НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО

Факультет інформатики та обчислювальної техніки
(назва факультету, інституту)

Кафедра автоматизованих систем обробки інформації і управління
(назва кафедри)

"На правах рукопису"
УДК 519.8

«До захисту допущено»
Завідувач кафедри

_____ О.А.Павлов
(підпис) (ініціали, прізвище)

“____” _____ 20*19* р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ

на здобуття ступеня магістра

за спеціальністю *126 Інформаційні системи та технології*
(код та назва спеціальності)

ОПП *Інформаційні управляючі системи та технології*
(код та назва спеціалізації)

на тему: *Інтелектуалізація обчислень для задач розрахунку міцності конструкцій*

Виконала: студентка *VI* курсу групи ІС-82мп
(шифр групи)

Марочканич Олександр Романович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник *с.н.с., проф., д.ф-м.н. Хіміч О.М.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант *доц.,к.т.н. Жданова О.Г.* _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент *к.т.н., доцент ТК, Лісовиченко О.І.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

РЕФЕРАТ

Магістерська дисертація: 103 с., 18 рис., 24 табл., 1 додаток, 37 джерел.

Актуальність теми. Зараз одним з основних етапів при дослідженні об’єктів, явищ і процесів різної природи є математичне моделювання і пов’язаний ним комп’ютерний експеримент. Чисельні експерименти дають можливість, як планувати натурний експеримент, так і отримувати нові знання про ті процеси і явища для яких утруднений, або взагалі неможливий натурний експеримент. Велика кількість математичних моделей після виконання відповідних перетворень можуть бути описанні системами лінійних алгебраїчних рівнянь (СЛАР) з розрідженими матрицями.

Основними проблемами розробки ефективних паралельних алгоритмів є: аналіз структури матриці, або приведення її до відповідного вигляду, застосовуючи відповідні алгоритми перетворення; вибір ефективної декомпозиції даних; визначення ефективної кількості процесорних ядер і графічних прискорювачів, що використовуються для обчислень; визначення топології міжпроцесних зв’язків, яка зменшує кількість комунікацій і синхронізацій.

Саме для аналізу структури розрідженої матриці використовується нейрона мережа, яка дозволить виділити групи ненульових елементів, які можуть оброблятись незалежно. За результатами аналізу буде обиратись алгоритм, будуватись декомпозиція даних та обиратись кількість обчислювальних ядер, що забезпечить найкоротший час розрахунків для конкретної структури матриці.

Мета та завдання дослідження. Метою даної роботи є автоматизація проектування будівельних конструкцій, використовуючи нейронну мережу, дослідження паралельних методів та комп’ютерних алгоритмів для дослідження та розв’язування СЛАР з розрідженими матрицями нерегулярної структури та апробація алгоритмів при математичному моделюванні у прикладних задачах.

Досягнення мети базується на розробці оригінальних математичних методів та паралельних алгоритмів та їх реалізації у спеціальній системі, яка призначена для визначення та аналізу міцності будівельних конструкцій.

До завдань дослідження належать:

- розробка нейронної мережі для визначення типу розрідженої матриці нерегулярної структури та вибору оптимального алгоритму розвязку СЛАР;
- дослідження прямих паралельних алгоритмів для СЛАР з розрідженими матрицями нерегулярної структури з наближеними даними;
- апробація алгоритмів для математичного моделювання в прикладних задачах.

Об'єкт дослідження – математичні моделі, що описуються СЛАР з розрідженими матрицями нерегулярної структури.

Предмет дослідження – паралельні методи та комп'ютерні алгоритми знаходження розв'язку СЛАР з розрідженими матрицями нерегулярної структури.

Методи дослідження. У роботі застосовуються методи теорії матриць, лінійної алгебри, теорії графів, функціонального аналізу, теорії похибок, теорія нейронних мереж.

Наукова новизна: Наукова новизна отриманих результатів полягає у покращенні підходу вирішення задачі визначення міцності будівельних конструкцій шляхом автоматизації процесу вибору ефективного алгоритму за допомогою нейронної мережі, застосування якої дозволило значно зменшити час прорахунку міцності будівельної конструкції та оптимізації комп'ютерних ресурсів.

Публікації: За матеріалами дисертації було опубліковано 3 наукові роботи: 1 стаття та 2 тез доповідей на конференціях.

СЛАР, ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ, РОСПІЗНАВАННЯ ЗОБРАЖЕНЬ, НЕЙРОННА МЕРЕЖА, КЛАСИФІКАЦІЇ ДАНИХ, НЕНУЛЬОВІ ЕЛЕМЕНТИ

ABSTRACT

Master's Thesis: 103 pp., 18 figs., 24 tables, 1 appendix, 37 sources.

Actuality of theme. Now one of the main stages in the study of objects, phenomena and processes of different nature is mathematical modeling and the related computer experiment. Numerical experiments provide an opportunity both to plan a full-scale experiment and to gain new knowledge about those processes and phenomena that make it difficult or impossible to do a full-scale experiment. A large number of mathematical models after the corresponding transformations can be described by systems of linear algebraic equations (SLAE) with sparse matrices.

The main problems of developing effective parallel algorithms are: analysis of the structure of the matrix, or bringing it to the corresponding form, using appropriate conversion algorithms; choice of effective data decomposition; determining the effective number of processor cores and graphic accelerators used for calculations; definition of the interprocess communication topology, which reduces the number of communications and synchronizations.

It is precisely for analyzing the structure of a sparse matrix that a neural network is used which allows the selection of groups of non-zero elements that can be processed independently. According to the results of the analysis, the algorithm will be selected, the data decomposition will be built and the number of computational nuclei will be selected, which will provide the shortest calculation time for a specific matrix structure.

The purpose and objectives of the study. The purpose of this work is to automate the design of building structures using a neural network, the study of parallel methods and computer algorithms for the study and solution of SLAR with sparse matrices of irregular structure, and the testing of algorithms in mathematical modeling in applied problems.

The goal is based on the development of original mathematical methods and parallel algorithms and their implementation in a special system that is designed to determine and analyze the strength of building structures.

Research objectives include:

- development of a neural network to determine the type of sparse matrix of irregular structure and to choose the optimal algorithm for solving SLAR;

- investigation of direct parallel algorithms for SLAR with sparse irregular structure matrices with approximate data;
- development of algorithms and programs for investigating the validity of the solutions obtained by direct and iterative methods;
- approbation of algorithms for mathematical modeling in applied problems.

The object of study is mathematical models that describe SLAEs with sparse matrices of irregular structure.

The subject of the study are parallel methods and computer algorithms for finding the SLAE solution with sparse matrices of irregular structure.

Research methods. The methods of matrix theory, linear algebra, graph theory, functional analysis, error theory, neural network theory are applied.

Scientific novelty: The scientific novelty of the results obtained is to improve the approach of solving the problem of determining the strength of structures by automating the process of selecting an efficient algorithm using a neural network, which significantly reduced the time to calculate the strength of the structure and optimization of computer resources.

Publications: Based on the dissertation materials, 3 scientific papers were published: 1 article and 2 abstracts at conferences.

SLAE, PARALLEL CALCULATION, IMAGE RECOGNITION, NEURAL NETWORK, DATA CLASSIFICATION, NON-ZERO ELEMENTS

ЗМІСТ

ВСТУП

На сьогоднішній день розрахунок міцності конструкцій є однією з найбільш ресурсозатратних областей математичного моделювання. Основна ідея сучасного підходу – інтеграція в рамках єдиного програмно-технічного комплексу наступних задач (які в більшій або в меншій мірі характерні для більшості прикладних задач):

- об′ємне моделювання твердих тіл;
- отримання проектних рішень (як для елементів конструкцій, так і для конструкцій в цілому);
- обробка обчислювальної інформації;
- візуалізація результатів розрахунків;
- створення конструкторських креслень.

До найбільш повних (по рівню інтеграції етапів проектування) можна віднести так звані САД/CAM системи: IDEAS фірми CDRS, UNIGRAPHICS фірми Mc.DONNELL DUGLAS, ANSYS, DUCT5 фірми DELCAM та ін. Існує багато інших як вітчизняних, так і зарубіжних програмних продуктів з меншим ступенем інтеграції. Одними з найпоширеніших програмних продуктів вітчизняного виробництва для автоматизованого проектування і конструювання, чисельного дослідження міцності і стійкості будівельних конструкцій є, наприклад, програмний комплекс ЛІРА.

Зосередимось на етапі проектування. Для отримання проектних рішень необхідно сформувати розв’язуючу задачу, розв’язок якої отримують за допомогою відповідного чисельного методу. Але, не дивлячись на досить повне теоретичне обґрунтування чисельних методів, розв'язування прикладних задач в ряді випадків призводить до комп'ютерних розв'язків, які не мають фізичного змісту. Це відбувається через нехтування похибками вихідних даних, відмінності властивостей використовуваних математичних і машинних моделей задач, відмінності класичної та комп'ютерної арифметики, тощо. Тому актуальною проблемою є дослідження достовірності отримуваних за допомогою комп'ютера наближених розв’язків.

Особливу увагу слід приділити вивченню впливу наближених даних прикладної задачі на результати розрахунків. Адже майже всі вихідні дані прикладної задачі – фізичні константи, якими задаються механічні властивості матеріалу конструкції (наприклад, модуль Юнга, коефіцієнт Пуассона), геометричні розміри, навантаження, тощо задаються наближено, а в теоретичних обґрунтуваннях чисельних методів, які застосовуються для отримання розв’язків, неявно вважається, що дані задано точно.

При цьому важливим є аналіз тих багатовимірних взаємопов'язаних фізико-механічних процесів, які визначають несучу здатність окремих конструкційних елементів з урахуванням усього спектру експлуатаційного силового впливу, а також прогнозування необоротних змін в конструкції, що мають наслідком зародження і розвиток мікро-і макродефектів матеріалу.[1] Можливе розглядання декількох показників одночасно.

Складність задач розрахунку міцності конструкцій, які необхідно розв’язувати, високі вимоги до якості проектних рішень приводять, крім того, до великих порядків матриць, побудованих методом скінченних елементів (МСЕ), які можуть досягати десятків мільйонів, і вимагають великих обчислювальних ресурсів (зокрема пам'яті). Але ресурсів сучасних високопродуктивних однопроцесорних комп’ютерів недостатньо для розв’язування таких задач. Тому актуальною є проблема застосування паралельних комп’ютерів, зокрема MIMD-комп’ютерів.

Мета та завдання дослідження. Метою даної роботи є автоматизація проектування будівельних конструкцій, використовуючи нейронну мережу, дослідження паралельних методів та комп’ютерних алгоритмів для дослідження та розв’язування СЛАР з розрідженими матрицями нерегулярної структури та апробація алгоритмів при математичному моделюванні у прикладних задачах.

Досягнення мети базується на розробці оригінальних математичних методів та паралельних алгоритмів та їх реалізації у спеціальній системі, яка призначена для визначення та аналізу міцності будівельних конструкцій.

До завдань дослідження належать:

- розробка нейронної мережі для визначення типу розрідженої матриці нерегулярної структури та вибору оптимального алгоритму розвязку СЛАР;
- дослідження прямих паралельних алгоритмів для СЛАР з розрідженими матрицями нерегулярної структури з наближеними даними;
- апробація алгоритмів для математичного моделювання в прикладних задачах.

Об’єкт дослідження – математичні моделі, що описуються СЛАР з розрідженими матрицями нерегулярної структури.

Предмет дослідження – паралельні методи та комп’ютерні алгоритми знаходження розв’язку СЛАР з розрідженими матрицями нерегулярної структури.

Методи дослідження. У роботі застосовуються методи теорії матриць, лінійної алгебри, теорії графів, функціонального аналізу, теорії похибок, теорія нейронних мереж.

Наукова новизна: Наукова новизна отриманих результатів полягає у покращенні підходу вирішення задачі визначення міцності будівельних конструкцій шляхом автоматизації процесу вибору ефективного алгоритму за допомогою нейронної мережі, застосування якої дозволило значно зменшити час прорахунку міцності будівельної конструкції та оптимізації комп’ютерних ресурсів.

Вищенаведені проблеми можна вирішити за допомогою прикладного інтелектуального програмного забезпечення аналізу міцності конструкцій. Його концепція полягає в автоматизації процесів: вибору математичної моделі, побудови дискретної моделі, формування в комп’ютері відповідної машинної задачі, дослідження математичних властивостей комп’ютерної моделі задачі з наближеними даними, вибору паралельного алгоритму розв’язування в залежності від виявлених властивостей з урахуванням технічних можливостей комп'ютера, побудови топології із процесорів паралельного комп'ютера, одержання розв’язку задачі та дослідження його достовірності. При цьому забезпечується спілкування з кінцевим користувачем на мові його предметної області.

Статистичне прогнозування міцності та надійності відповідальних конструкцій є практичною альтернативою детермінованим методам проектування та експертного аналізу фактичного стану. Це, зокрема, дозволяє уникнути консервативного припущення про те, що більшість параметрів конструкції і властивостей матеріалу є відомими постійними, і враховувати невизначеність вхідних даних.

Що стосується оцінювання стану магістральних і технологічних трубопроводів під внутрішнім тиском, однією з характерних завдань є визначення міцності і надійності з урахуванням наявності експлуатаційних дефектів, таких як корозійні (ерозійні) втрати металу. Концентрація напруг в області стоншування стінки викликає збільшення схильності матеріалу до зародження і поширення руйнувань при порівняно низькій зовнішній навантаженості.

Існує ряд стандартних алгоритмів, що дозволяють оцінити рівень зниження міцності дефектного трубопровідного елемента (ТЕ) з виявленою в процесі дефектоскопії корозійно-ерозійної втратою металу [1-3]. Використовувані критерії граничного стану вимагають певного набору вхідних параметрів, таких як фізико-механічні властивості матеріалу, фактична геометрія конструкції, характеристики опору певного типу руйнування; крім того, зазвичай робиться припущення про однорідність матеріалу і його ізотропності.

Для зниження консервативності оцінки граничного стану ТЕ з корозійними пошкодженнями з урахуванням стохастичного просторового розподілу властивостей матеріалу можуть бути використані методи статистичної теорії міцності, найбільш простим з яких є метод Монте-Карло. Але реалізація цього підходу вимагає точного опису докритичного і критичного руйнування ТЕ під дією експлуатаційного навантаження (внутрішнього тиску P), а також відповідного високопродуктивного програмного забезпечення для її реалізації.

Запропоновано комплексний підхід чисельного прогнозування міцності та надійності трубопроводів з виявленими дефектами корозійної втрати металу з просторово неоднорідними властивостями за допомогою методу Монте-Карло для

реалізації статистичного аналізу міцності поряд з кінцево-елементних описом стану конкретної конструкції.

Для побудови ефективних алгоритмів оцінки статистичної міцності відповідальних трубопровідних елементів (ТЕ) і посудин тиску з урахуванням технологічних процесів їх монтажу і експлуатації необхідно вирішення низки завдань, а саме:

- побудова скінчено-елементної моделі напружено-деформованого і граничного станів конструкції в процесі монтажного зварювання і навантаження внутрішнім тиском;
- розробка алгоритму розрахунку ймовірності руйнування ТЕ з виявленими дефектами локального стоншування стінки з урахуванням стохастичності вхідних даних за методом Монте-Карло;
- реалізація високопродуктивних програмних засобів для комп'ютерів багатопроцесорної і гібридної архітектури.

Стосовно задачі прогнозування граничного стану ТЕ з поверхневим дефектом локального стоншування стінки показав свою ефективність скінчено-елементний опис напружено-деформованого стану металу в умовах зовнішнього термосилового впливу наряду з моделюванням зародження і розвитку в'язкого руйнування. Так, кінетика температурного поля при монтажному зварюванні, яка моделювалася для обліку залишкового післязварного стану, оцінювалася шляхом чисельного розв'язання нестаціонарного рівняння теплопровідності під дією нормально розподіленого поверхневого джерела тепла (що є загальноприйнятим, зокрема, при розгляді технологічного процесу дугового зварювання).

1 ОГЛЯД СУЧАСНОГО СТАНУ ПРОБЛЕМИ

1.1 Огляд літератури

Відомо, що 80-90% науково-технічних задач зводяться до розв’язування задач обчислювальної математики, значну частину яких складають завдання, пов’язані з розв’язуванням та дослідженням СЛАР. Тому розробка ефективних підходів знаходження розв’язків систем рівнянь є надзвичайно актуальною проблемою, вирішенню якої приділяють чимало уваги.[2]

Значна кількість досліджень проводиться для систем з розрідженими матрицями, з якими пов’язано немало практичних застосувань. На даний момент найбільш повно питання, пов’язані з розв’язуванням СЛАР з розрідженими матрицями, висвітлено у вітчизняних [1, 2, 3, 4, 5, 6, 7] та зарубіжних працях [8, 9, 10, 11, 12, 13, 14].

Монографія [1] висвітлює ряд фундаментальних положень роботи з розрідженими матрицями. У монографії [2] розглянуто широкий спектр питань, пов’язаних як з методами розв’язування СЛАР, так і з допоміжними проблемами, такими, як формати зберігання даних чи алгоритми попередньої оптимізації впорядкування. У роботах [2, 3] проведено ґрунтовні дослідження питань пов’язаних з форматами та попередньою обробкою розріджених даних. Монографія [15] вважається однією з фундаментальних щодо методів розв’язування СЛАР з додатньо визначеними розрідженими матрицями великих порядків. Монографія [16] присвячена розробці прямих методів. У роботі [10] подано надзвичайно широкий опис з теоретичним обґрунтуванням різних класів ітераційних методів розв’язування СЛАР. Монографія [5] описує ітераційні методи з передобумовленням з допомогою неповної факторизації. Роботи [9, 18] дають змогу провести порівняльний аналіз методів для щільних та розріджених матриць. У [4] запропоновано ефективні обчислювальні схеми для СЛАР з λ-матрицями. Отже, на даний момент існує потужна методична база загальних методів. Проте, як правило, на практиці нерідко задачі мають особливості, використання інформації про які дозволяє будувати ще більш ефективні схеми.[14]

При вирішенні задачі розв’язування СЛАР з розрідженими матрицями можна скористатися одним з двох підходів. Перший полягає у використанні прямих методів.[15] Цей напрямок представлено досить широко як у новітніх, так і в фундаментальних розробках, другий підхід пропонує ітераційні методи, розробці яких присвячено ряд робіт.

Складність розв’язуваних на комп’ютері задач та об’єм інформації, яку необхідно обробляти при розв’язуванні різних класів науково-технічних задач, у тому числі й у випадку СЛАР, постійно зростають. Попри швидке збільшення продуктивності комп’ютерів за рахунок розвитку елементної бази, комп’ютери з одним потоком команд і даних не завжди здатні забезпечити необхідні обчислювальні ресурси.[9]

Одним з резервів підвищення продуктивності обчислювальної техніки є використання паралельних комп’ютерів. На сьогодні паралельні комп’ютери виступають основним засобом для природничих наукових досліджень.[16] Аналіз швидкодії комп’ютерів, які використовуються у різноманітних областях діяльності, показує, що перші п’ятсот місць за виробничими показниками займають багатопроцесорні системи. Вони широко використовуються при моделюванні складних процесів та явищ в геології, хімії, фармакології, біології, медицині, метеорології та інших галузях сучасних природничих наук. Досить широке застосування паралельні комп’ютери знаходять у машинобудуванні, будівництві, екології, ядерній енергетиці. Лише врахування архітектури паралельних комп’ютерів, їх функціональних можливостей та технічних характеристик комунікаційної мережі дають змогу отримати високу експлуатаційну продуктивність мультипроцесорних комп’ютерів. Тому розробка нових алгоритмів або розпаралелювання існуючих для реалізації на паралельних комп’ютерах певного класу виступають актуальними задачами.[16]

Окремо значну групу досліджень становлять розробки алгоритмів для ітераційних методів розв’язування СЛАР. Теми розробки та аналізу ітераційних методів розв’язування СЛАР з розрідженими матрицями є досить актуальними, що

спричинене їхньою близькістю до прикладних задач. Саме тому на математичних форумах існують цілі розділи, присвячені цим питанням: проблемам побудови передобумовлювача, вибору методу розв’язування СЛАР та іншим.

Найбільш фундаментальний огляд питань, пов’язаних з ітераційними методами розв’язування СЛАР з розрідженими матрицями, подано в монографії [17]. У ній наведено розгорнутий огляд базових понять лінійної алгебри, методи дискретизації, теорію розріджених матриць, повний перелік різних класів ітераційних методів, способи передобумовлення та питання паралельної реалізації. У доповнення монографії автори опублікували ряд статей: передобумовлення на основі неповної факторизації [12], різні способи покращення неповної факторизації [11], побудова відповідного програмного забезпечення [13] та інші (всього близько 15 базових питань, наведених в монографії).

Проте, не зважаючи на значну кількість досліджень, присвячених побудові паралельних алгоритмів швидкозбіжних ітераційних методів розв’язування СЛАР, проблема лишається актуальною. Досі не розроблено методу, який би ефективно розв’язував задачу з матрицею будь-якої структури. Навіть найпопулярніший сьогодні підхід, пов'язаний з побудовою передобумовлювача на основі методу неповної факторизації, знаходить розв’язки не для всіх класів задач. Крім того, у ряді випадків ефективність розпаралелювання вдається досягти лише на невеликій кількості процесів.[16]

Навіть з цього короткого огляду літератури щодо паралельних обчислень видно, яка велика увага приділяється проблемі побудові ефективних паралельних алгоритмів розв’язування СЛАР з розрідженими матрицями. Проте, як показує досвід, необхідною умовою створення ефективних паралельних алгоритмів і програм для розв’язування СЛАР з розрідженими матрицями є врахування архітектури паралельних комп’ютерів та особливостей структури розрідженої матриці з метою забезпечення високої ефективності алгоритмів та збалансованості обчислювальних процесів.[15]

Тому до завдань магістерської роботи внесено розробку ефективних паралельних алгоритмів розв’язування систем з розрідженими матрицями, зберігання та обробки матриць для MIMD-комп’ютерів. Такі задачі виникають, наприклад, при математичному моделюванні міцністного аналізу конструкцій.

1.2 Огляд існуючих рішень

Окрім використання багатопроцесорних систем ще одним ефективним підходом щодо прискорення обчислень виступають графічні процесори, розвиток яких набув особливо швидких темпів протягом останніх років. Для розробки програм з використанням графічних процесорів розроблено програмні засоби, у тому числі для них створено аналоги математичних бібліотек типу BLAS – CUBLAS, які дозволяють значно скоротити час виконання обчислювальних операцій. Отже, одним з підходів побудови ефективних програм є розробка паралельних алгоритмів для гібридних систем, які окрім багатопроцесорного комп’ютера MIMD-архітектури включають графічні процесори SIMD-архітектури.

Проблеми розвитку прикладного програмного забезпечення для розв’язування СЛАР та супутніх задач лінійної алгебри знаходяться у центрі уваги багатьох спеціалістів з часів розробки перших комп’ютерів. Появу в кінці 60-х алгол-процедур, розроблених Вілкінсоном та Райншем, можна віднести до початку створення математичного забезпечення лінійної алгебри. Алгол-процедури ввійшли до складу бібліотек прикладних програм WR. Ці процедури у подальшому лягли в основу багатьох бібліотек та пакетів прикладних програм з лінійної алгебри.[15]

Серед найвідоміших бібліотек, які витримали випробовування часом – бібліотека SSP фірми IBM, Boeing Library, Bell Laboratories, бібліотека IMSL (International Mathematical and Statistical Libraries), бібліотека NAG (Numerical Algorithms Group). Великою кількістю підпрограм у них реалізовано клас задач з лінійної алгебри.

Також варто зазначити, що останнім часом завдяки постійному вдосконаленню обчислювальних характеристик багатопроцесорних комплексів

набувають актуальності питання, пов’язані з розробкою паралельних методів розв’язування систем рівнянь. У цьому напрямку розроблено чимало нових паралельних програмних засобів для розв’язування власне СЛАР. Більшість з них орієнтовані на розв’язування певних класів задач, що впливає на набір методів, які вони пропонують.[17] Наприклад, в пакетах BlockSolver95 і SuperLU реалізовано прямі методи розв’язування СЛАР. Ряд інших програмних комплексів, наприклад, pARMS, пропонують ітераційний підхід до вирішення проблеми. Оскільки більшість задач мають прикладний характер, значна частина програм орієнтована на СЛАР певного вигляду. Так, у задачах у будівельній, машинобудівельній галузі матриці коефіцієнтів систем зазвичай мають розріджену структуру.[19] Саме для матриць розрідженої структури розроблено низку як прямих (наприклад, у пакетах MUMPS, PSPASES), так і ітераційних (наприклад, пакети PSparslib, SPARSKIT) методів.

Актуальність теми розробки програмного забезпечення для розв’язування СЛАР підтверджується також активним обговоренням різного плану питань, пов’язаних з побудовою та програмною реалізацією прямих та ітераційних алгоритмів розв’язування СЛАР.

Методи розв’язування СЛАР знаходять своє застосування в електроніці, будівництві, машинобудуванні та багатьох інших прикладних областях. А, оскільки кожна окрема галузь застосування висуває свої вимоги до задач, то, не дивлячись на досить широкий вибір готових рішень, виникає потреба у розробці спеціалізованого програмного забезпечення, орієнтованого на певний клас задач.[19]

Ще одним важливим аспектом проблеми є достовірність результатів. Адже існуюче чисельне програмне забезпечення для розв’язування СЛАР створювалися з припущенням того, що вихідні дані задані точно. Але при розв’язуванні прикладних задач їхні математичні моделі мають, як правило, наближені дані та похибки, пов’язані з обмеженою розрядністю чисел, що зберігаються в пам’яті комп’ютера і в результаті заокруглень під час виконання арифметичних операцій.[20]

Отже, основним завданням дисертаційної роботи є розробка та апробація паралельних алгоритмів прямих та ітераційних методів розв’язування СЛАР з розрідженими матрицями нерегулярної структури.[15] Складовими підзадачами виступають розробка та дослідження форматів зберігання розріджених матриць, алгоритми оптимізації заповнення, питання вибору способу розподілу даних між процесами з урахуванням особливостей матриць, організація розподілених обчислень. Окремо для ітераційних методів постають питання побудови передобумовлювача та вибору параметрів обчислювальної схеми, розробки критерію закінчення ітераційного процесу. Також з метою мінімізації часу розрахунків розглядаються способи оптимізації алгоритмів, пов’язані з розширеними можливостями апаратного забезпечення, зокрема з графічними процесорами.

1.3 Проблеми розробки паралельних алгоритмів

На сьогоднішній день суперкомп’ютери паралельної архітектури відіграють роль основного обчислювального засобу в різноманітних предметних областях. Найвагомішою причиною зростання частини розрахунків на паралельних комп’ютерах є те, що збільшення тактової частоти існуючих процесів досягнуло критичної межі і основним резервом підвищення продуктивності комп’ютерів стало розпаралелювання обчислень.

Вважатимемо, що паралельний обчислювальний комплекс має такі складові:

- хост-комп’ютер для здійснення керування використання багатопроцесного обчислювального ресурсу, проведення загальносистемного моніторингу, комунікації з термінальними мережами користувачів, візуалізації розв’язків задач та реалізації тієї частини обчислювального процесу, яка не розпаралелюється;
- обробляюча частина, що містить обчислювальні вузли для розв’язування задачі з паралельною організацією обчислень; вона є однорідною масштабованою системою, яка складається з багатьох високопродуктивних

процесів з власною оперативною та дисковою пам’яттю, об’єднаних комунікаційним середовищем міжпроцесної взаємодії;
- дискове сховище для зберігання програмних модулів, великорозмірних даних та результатів обчислень;
- комунікаційні середовища та комутаційне середовище, призначені для ефективної взаємодії обчислювальних вузлів при проведенні розрахунків.

Операційна система хост-комп’ютера повинна забезпечувати виконання ряду завдань, таких, як запуск програми на хост-комп’ютері, формування завдання і запуск процесу розв’язування задачі на вибраній кількості процесів, моніторинг виконуваних завдань, збереження і візуалізація протоколів паралельних розрахунків, адміністрування доступних частин розподіленої файлової системи. Також має бути встановлено відповідне середовище міжпроцесної взаємодії та компілятор, що підтримує мову програмування, на якій написано виконувану програму.

З іншого боку, для створення алгоритмів з ефективним розпаралелюванням обчислень, необхідно враховувати особливості архітектури паралельних комп’ютерів та їхніх міжпроцесних з’єднань. При цьому слід брати до уваги властивості реальних обчислювальних ресурсів: обмежену кількість процесів, час обмінів, синхронізацію, об’єм пам’яті. Тому серед основних вимог до програм, написаних для виконання на MIMD-комп’ютерах, виділяють [12]:

- паралелізм – здатність виконання програми багатьма процесами одночасного;
- масштабованість – забезпечення можливості виконання програми з використанням різної кількості процесів;
- локальність – така організація обчислень, при якій звернення до локальних даних відбувається значно частіше, ніж до віддалених.

Ефективність алгоритмів у значній мірі визначається схемою розподілу вихідних даних між процесами. Наприклад, найпростішим способом розподілу є блочний, при якому матриця розподіляється на декілька рівних блоків, кількість яких дорівнює кількості процесів. Недоліком способу є так званий ефект Гайдна,

згідно з яким при обробці матриці процеси один за інших виходять з обчислень. Повністю уникнути цього дозволяє рядково-циклічний спосіб, при якому матриця між процесами розподіляється циклічно рядками. Слабкою стороною цього підходу є значна кількість синхронізацій між процесами. Способом, що об'єднує переваги наведених методів є блочно-циклічний, при якому матриця розподіляється між процесами блоками, але циклічно. При використанні кеш-пам'яті, що, як правило має невеликі об'єми, ефективним є застосування двовимірного блочно-циклічного розподілу, при якому блоки містять не цілі рядки матриці, а їх частини. Цим способом навіть матриці великих порядків можна розподілити таким чином, щоб обробка окремого блоку відбувалася у межах кеш-пам'яті.

Також не менш важливою є відповідність алгоритму розв'язування задачі архітектурі комп'ютера та особливості топології міжпроцесних зв'язків. Тому на значення коефіцієнтів ефективності EP та прискорення SP впливає ряд параметрів, пов'язаних з характеристиками MIMD-комп'ютера. Ці коефіцієнти визначаються, як [12]

$$Sp = T1 / Tp, Ep = Sp / p,$$

де p – кількість процесів, що використовуються для розв'язування задачі, Tp – час розв'язування задачі на MIMD-комп'ютері з використанням p процесів, T1 – час розв'язування тієї ж задачі на гіпотетичному послідовному комп'ютері зі швидкодією одного процесу та оперативною пам'яттю, яка рівна сумарній пам'яті, що використовується p процесами.

До величин T1 і Tp входять з коефіцієнтами, що залежать від конкретного алгоритму, такі складові, як t – середній час виконання однієї арифметичної операції, to – час, необхідний для обміну одним машинним словом між двома процесами, tc – час, який потрібен для встановлення зв'язків між двома процесами.

При побудові паралельних алгоритмів вважається, що необхідна для реалізації обчислювального алгоритму інформація зберігається і обробляється в оперативній пам'яті гіпотетичного послідовного комп'ютера або ж у сумарній пам'яті MIMD-

комп’ютера, на якому виконуються p процесів, тобто обчислювальний процес здійснюється без використання зовнішньої пам’яті.

Також на ефективність реалізації паралельного алгоритму впливає використання різних способів обміну даними між процесами. Процеси можуть взаємодіяти попарно за допомогою обмінів типу «точка-точка» або групою з використанням колективних обмінів. До найбільш часто використовуваних належать:

- мультирозсилка, за якої один з процесів взаємодіючої групи розсилає дані всі процесам цієї групи;
- мультизбірка числа, при якій усі процеси взаємодіючої групи передають рівні порції даних одному з процесів цієї групи; при цьому над відповідними компонентами даних від різних процесів виконуються операції зведення, наприклад, додавання, вибір максимального або мінімального значення і тому подібні;
- мультизбірка вектора, під час якої всі процеси взаємодіючої групи передають порції даних одному з процесів групи, в якому з прийнятих даних формується новий вектор.

Слід також зазначити, що паралельні обчислення можуть бути ефективно використані для розв’язування СЛАР з розрідженими матрицями. У цьому випадку додається ще одна можливість щодо покращення ефективності алгоритму, суть якої полягає у вдалому виборі формату зберігання даних та оптимізації заповнення матриці. Тобто, оскільки при обробці розрідженої матриці операції виконуються лише з ненульовими елементами, то при використанні певного алгоритму впорядкування можна розмістити їх таким чином, щоб зменшити кількість міжпроцесорних обмінів і одночасно забезпечити балансування процесів.

2 ПОСТАНОВКА ТА ФОРМАЛІЗАЦІЯ ЗАДАЧІ

2.1 Математичні постановки задач.

Математично задачі розрахунку міцності конструкцій з використанням принципу можливих зміщень можуть бути поставлені у вигляді варіаційних задач: необхідно знайти вектор-функцію $u \in U_0$, яка для будь-якої вектор-функції $v \in U_0$ (для будь-якого можливого переміщення) задовольняє одну з інтегральних тотожностей [21]:

– для статичної задачі

$$\boldsymbol{a}(u,v) = \boldsymbol{l}(f,v); \qquad (2.1)$$

– для динамічної задачі

$$\boldsymbol{a}(u,v) + \boldsymbol{b}(u'',v) + \boldsymbol{c}(u',v) = \boldsymbol{l}(f,v), \quad u(t_0) = u^{(0)}, \; u'(t_0) = u^{(1)}; \qquad (2.2)$$

– для задачі на власні коливання

$$\boldsymbol{a}(u,v) = \lambda \boldsymbol{b}(u,v), \qquad (2.3)$$

де U_0 – нескінченновимірний функціональний простір можливих переміщень, симетричні білінійні функціонали $\boldsymbol{a}(u,v)$, $\boldsymbol{b}(u'',v)$, $\boldsymbol{c}(u',v)$ пропорційні відповідно потенційній, кінетичній енергіям деформації і енергії гальмування, а лінійний функціонал $\boldsymbol{l}(f,v)$ пропорційний роботі прикладених (зовнішніх) зусиль при навантаженні (через u' позначено першу похідну вектор-функції $u(t,\chi)$ за часом, а через u'' – другу). [21]

Тут розглядаються тільки лінійні задачі, оскільки, як правило, розв’язування нелінійної задачі розрахунку міцності конструкції зводиться до розв’язування послідовності лінійних задач.

Теоретичною основою більшості програмних засобів для розрахунку міцності конструкцій є метод скінченних елементів (МСЕ), реалізований у формі зміщень. Вибір саме цієї форми пояснюється простотою її алгоритмізації і фізичної інтерпретації, наявністю єдиних методів побудови матриць жорсткості і векторів

навантажень для різних типів скінченних елементів, можливістю врахування довільних граничних умов і складної геометрії конструкції, що розраховується. [21]

Дискретизація варіаційної задачі методом скінченних елементів, який є типовим представником проекційних методів, полягає в заміні нескінченновимірного простору допустимих функцій U_0 його скінченновимірним підпростором U_0^{h}. Вектор-функції з підпростору U_0^{h} можуть бути представлені у вигляді лінійної комбінації базисних вектор-функцій, які задовольняють головні (кінетичні) граничні умови [22]:

$$u^h(\cdot) = \sum_{i=1}^{n} x_i \varphi_i(\cdot), \quad (2.4)$$

де φ_i (i = 1, 2,…, n) – згаданий вище базис U_0^{h}, звичайно кусково-поліноміальний.

Тоді білінійні і лінійний функціонали з (1)-(3) в скінченновимірному просторі U_0^{h} можна записати у вигляді відповідно білінійних і лінійної форм коефіцієнтів x_i в представленні (4), які надалі називатимемо вузловими параметрами,

$$\boldsymbol{a}(u^h,v^h) \equiv y^{\mathrm{T}}Ax, \quad \boldsymbol{b}(u^h,v^h) \equiv y^{\mathrm{T}}Bx, \quad \boldsymbol{c}(u^h,v^h) \equiv y^{\mathrm{T}}Cx, \quad \boldsymbol{l}(f,v^h) \equiv y^{\mathrm{T}}b, \qquad (2.5)$$

де x і y – вектори вузлових параметрів відповідно функцій u^h і v^h, а елементи матриць жорсткості (A), мас (B), демпфування (C) і вектора навантажень (b) обчислюються за формулами (i, j = 1, 2, …, n) [22]:

$$a_{ij} = \boldsymbol{a}(\varphi_i, \varphi_j), \quad b_{ij} = \boldsymbol{b}(\varphi_i, \varphi_j), \quad c_{ij} = \boldsymbol{c}(\varphi_i, \varphi_j), \quad b_i = \boldsymbol{l}(f, \varphi_i). \qquad (2.6)$$

Таким чином, враховуючи в (5) довільність вектора y, отримуємо відповідні дискретні задачі [21, 22]:

– для статичної задачі (1) СЛАР

$$Ax = b; \qquad (2.7)$$

– для динамічної задачі (2) – задачу з початковими умовами

$$Bx''(t) + Cx'(t) + Ax(t) = b(t), \quad x(t_0) = x^{(0)}, \; x'(t_0) = x^{(1)}; \qquad (2.8)$$

– для задачі на власні коливання (2.3) – узагальнену алгебраїчну проблему власних значень

$$Ax = \lambda^h Bx. \quad (2.9)$$

2.2 Формування і розв’язування дискретних задач.

Враховуючи, що базисні функції підпростору МСЕ U_0^h можна вибрати так, щоб вони були відмінними від нуля лише на декількох скінченних елементах, структура матриць з (2.5) є в загальному випадку розрідженою і визначається нумерацією вузлових параметрів, яка у свою чергу залежить від нумерації вузлів скінченно-елементної сітки. Тобто матриці задач (2.7)-(2.9) є стрічковими, профільними, блочно-діагональними з обрамленням і т.п. Крім того, з погляду комп'ютерного розв’язування цих задач істотне значення має те, що матриці дискретних задач є симетричними, додатно означеними або додатно напівозначеними, а також що порядок цих матриць складає $O(10^5) – O(10^7)$. [23]

Для побудови простору МСЕ U_0^h вихідна область розбивається на підобласті-елементи. На кожному елементі функції з простору МСЕ є поліномами заданої степені, коефіцієнти яких визначаються через вузлові параметри – значення функцій (а у ряді випадків і їх похідних) у вузлах елемента (точки у вершинах, на сторонах, гранях або усередині елемента). [24] Для вузлових параметрів і базисних функцій справедливі такі співвідношення

$$L_j(\varphi_i) = \delta_{ij}, \quad (2.10)$$

де δ_{ij} – символ Кронекера, а результатом операції $L_j(\varphi_i)$ є значення компоненти вектор-функції φ_i для вузлового параметра L_j.

Важливою перевагою МСЕ, враховуючи (2.4) і (2.10), є те, що елементи матриць і векторів правих частин задач (2.7)-(2.9) отримують підсумовуванням відповідних елементів відповідних матриць і векторів навантажень, побудованих для окремих скінченних елементів. Така властивість матриць і векторів дискретних задач МСЕ дозволяє ефективно розпаралелювати процес формування цих матриць і векторів. [24] При цьому можна розпаралелювати тільки обчислення та розподіл між

паралельними процесами відповідно до вимог методу, який використовуватиметься для розв’язування дискретної задачі, елементів глобальних матриць і векторів дискретних задач, використовуючи обчислені раніше матриці і вектори окремих скінченних елементів, або розпаралелювати також обчислення цих матриць і векторів окремих скінченних елементів.

З метою мінімізації кількості арифметичних операцій при розв’язуванні дискретних задач, а також необхідної оперативної і зовнішньої пам'яті, враховуючи розріджену структуру матриць задач (2.7)-(2.9), в більшості випадків проводиться переупорядковування невідомих задачі. [25] Залежно від критерію мінімізації – ширини стрічки матриці, її профілю, загальної кількості арифметичних операцій при трикутному розвиненні матриці і т. д. – використовуються різні методи переупорядковування, наприклад, зворотний алгоритм Катхілла-Маккі, фактор-дерев, мінімальної степені. Конкретних рекомендацій для вибору методу впорядкування дати не можна, тому що ефективність того або іншого алгоритму суттєво залежить від первинної структури конкретної матриці. [25]

Для розв’язування СЛАР (2.7) – вихідної або переупорядкованої – на MIMD-комп'ютері можуть бути застосовані паралельні алгоритми для дослідження і розв’язування СЛАР з симетричними матрицями, які використовують LDL^T-розвинення методу Холецького або U^TDU-розвинення методу Гауса. [26] Залежно від ширини і заповнення стрічки матриці використовуються:

– блочні паралельні алгоритми для вузьких стрічкових матриць і блочно-діагональних матриць з обрамленням;

– одновимірні блочно-циклічні паралельні алгоритми для дослідження і розв’язування СЛАР із стрічковими або профільними матрицями, в тому числі з матрицями так званої «хмарочосної» структури (у них не беруть участь в обчисленнях внутрішні елементи профілю матриці, які залишаються нульовими в процесі розвинення матриці). [26]

В напівдискретному МСЕ наближений розв’язок шукається у вигляді (2.4), де коефіцієнти x_i є функціями часу t. В результаті отримуємо систему звичайних

диференціальних рівнянь другого порядку з початковими умовами (8), де *x(t), x*$^{(0)}$, *x*$^{(1)}$ – вектори з елементами $x_i(t)$, $x_i^{(0)} = L_i(u^{(0)}), \quad x_i^{(1)} = L_i(u^{(1)})$.

В переважній більшості програмних засобів розрахунку міцності конструкцій система (8) розв'язується методом розвинення за формами власних коливань. Якщо $\lambda_k^h,\ z_k^h \ \ (z_k^{h\mathsf{T}} B z_k^h = 1,\ k = 1, 2, \ldots, n)$ – розв’язок алгебраїчної проблеми власних значень (9), то, вважаючи в (8) $x(t) = \sum_{k=1}^{n} y_k(t) z_k^h$, отримаємо (при *B*-ортогональності векторів z_k^h та деяких припущеннях щодо матриці демпфування *C*) систему, яка розпадається на незалежні відносно $y_i(t)$ рівняння [27]:

$$y_k''(t) + 2\xi_k \omega_k y_k'(t) + \omega_k^2 y_k(t) = P_k(t), \quad t > t_0,$$
$$y_k(t_0) = y_k^{(0)}, \quad y_k'(t_0) = y_k^{(1)}, \qquad (2.11)$$

де $\omega_i = \lambda_i^{-0,5}$, $0 < \xi_k < 1$, $P_k(t) = z_k^{h\mathsf{T}} b(t)$, $y_k^{(0)} = x^{(0)\mathsf{T}} B z_k^h$, $y_k^{(1)} = x^{(1)\mathsf{T}} B z_k^h$, *k* = 1, 2, …, *n*. Розв’язки задач (11) мають вигляд:

$$y_k(t) = e^{-\xi_k \omega_k t}\left(\frac{y_k^{(1)} + y_k^{(0)} \xi_k \omega_k}{w_k} \sin \overline{\omega}_k t + y_k^{(0)} \cos \overline{\omega}_k t\right) + \frac{1}{\overline{\omega}_k} \int_0^t P_k(\tau) e^{-\xi_k \omega_k (t-\tau)} \sin \overline{\omega}_k (t-\tau) d\tau,$$

де $\overline{\omega}_k = \omega_k \sqrt{1 - \xi_k}$. Аналіз цих розв’язків свідчить про те, що істотний внесок в розв’язок *x*(*t*) задачі (8) дають лише близько 10-20 складових, які відповідають мінімальним власним значенням. [27]

Існує ряд випадків навантажень, коли можливе точне обчислення розв’язків задач (11), наприклад, для вітрового, сейсмічного або гармонійного навантажень. В решті випадків розв’язки $y_k(t)$ обчислюються чисельно, наприклад, методом скінченних різниць по схемі Ньюмарка. Також задачі Коші (2.11) можуть розв'язуватися методом Рунге-Кутта четвертого порядку, орієнтованим на розв’язування систем звичайних диференціальних рівнянь другого порядку. [28]

Основний висновок в контексті задач дисертаційного дослідження: міцністний розрахунок конструкцій як для статичного, так і динамічного аналізу конструкцій

(лінійного і нелінійного) зводиться до дослідження і розв’язування СЛАР, а саме до застосування розроблених в попередніх розділах паралельних алгоритмів для систем з симетричними додатно визначеними матрицями нерегулярної розрідженої структури в умовах наближених даних.

3 АЛГОРИТМИ РОЗВ'ЯЗУВАННЯ ЗАДАЧІ

3.1 Дрібно-плитковий гібридний алгоритм LL^{T} розвинення блочно-діагональної матриці з обрамленням

Розглянемо дрібно-плитковий гібридний алгоритм факторизації розрідженої блочно-діагональної матриці з обрамленням A. Даний алгоритм краще враховує профільну, або розріджену структуру діагональних блоків, блоків обрамлення. [29]

Розіб'ємо матрицю A на блоки розмірністю $c \times c$.

Далі для факторизації блочно-діагональної матриці застосуємо алгоритм запропонований в [18] для щільних матриць.

Для факторизації матриці на $к$-ому кроці використаємо наступне співвідношення [30]:

$$A^{\kappa} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22}^T \end{pmatrix} \tag{3.1}$$

де розміри блоків A_{11} – $c \times c$, A_{21} – (n-kc)с, A_{22} – (n-kc)(n-kc), блоки A_{21} та A_{22} враховують структуру блоків D_{ii}, C_{pi}, D_{pp}.

Звідси отримаємо алгоритм, за яким проводиться розвинення на $к$ кроці:

$$A_{11} = L_{11} * L_{11}^T; \tag{3.2}$$

$$L_{21} = A_{21} * \left(L_{11}^T\right)^{-1}; \tag{3.3}$$

$$\tilde{A}_{22} = A_{22} - L_{21} * L_{21}^T. \tag{3.4}$$

Зазначимо, що реалізація (3.2)-(3.4) на кожному кроці модифікує тільки блоки D_{ii}, C_{pi}, $i = \overline{1, p-1}$, D_{pp}. [30]

Для реалізації алгоритму будемо використовувати розподіл, який був вище запропонований. Тобто в GPU(i) $i = \overline{1, p-1}$ містяться блоки D_{ii}, C_{pi} та блок $A_{pp}^{(i)}$ того ж розміру, що й D_{pp}; в GPU(p) зберігається блок D_{pp}.

На рис. 3.1 показано блочний розподіл даних на *k*-му кроці факторизації блочно-діагональної матриці з обрамленням, враховуючи вище запропоновану декомпозицію.

Паралелізація обчислень трикутної факторизації полягає в тому, що розвинення блоків A_{11} та модифікація A_{21} та A_{22} може здійснюватись незалежно у кожному CPU(i) та GPU(i) $i=\overline{1,p-1}$.

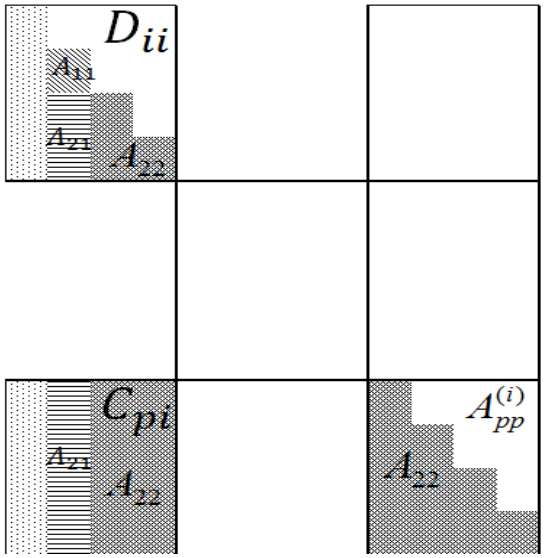


Рисунок 3.1 – Декомпозиція даних на GPU(i)

На кожному кроці у всіх парах CPU(i) та GPU(i) $i=\overline{1,p-1}$ виконуємо:

– у CPU(i), $i=\overline{1,p-1}$ факторизуємо A_{11} із D_{ii}:

$$A_{11} = L_{11} * L_{11}^{T};$$

– у GPU(i), $i=\overline{1,p-1}$ модифікуємо стовпчик блоків L_{21}:

$$L_{21} = A_{21} (L_{11}^{T})^{-1};$$

– у GPU(i), $i=\overline{1,p-1}$ модифікуємо блоки матриці A22 з $A_{pp}^{(i)}$ за формулою:

$$\tilde{A}_{22} = A_{22} - L_{21}L_{21}^{T};$$

– у CPU(p), використовуючи мультизбирання, модифікуємо D_{pp}:

$$D_{pp}^{*} = D_{pp} - \sum_{i=1}^{p-1} A_{pp}^{(i)}.$$

Факторизуємо блок D_{pp}^{*}, тим самим завершуючи процес факторизації матриці.

Будемо вважати, що порядки всіх діагональних блоків приблизно рівні

$$q_i \approx q = \frac{n-s}{p-1},$$

де s – порядок останнього діагонального блоку. Тоді справедливі наступні теореми.

Теорема 3.1. Кількість операцій, що виконуються на 1 GPU для знаходження факторизації розрідженої блочно-діагональної матриці з обрамленням оцінюється величиною [31]

$$N_p \approx \frac{q^3}{3} + sq^2 = \frac{q^2}{3}(q + 3s).$$

Теорема 3.2. Прискорення дрібно-плиткового гібридного алгоритму LL^{T}-розвинення розрідженої блочно-діагональної матриці з обрамленням A, становить [31]:

$$S_p \approx (p-1)\left(1 + \frac{3(p-1)s^2}{2q^2(q+3s)}\left(\tau_{opp} + \left(\frac{2p}{(p-1)} + \frac{4qc}{s^2} + \frac{4c}{(p-1)s}\right)\tau_{opg}\right)\right)^{-1},$$

де c – розмір плитки.

Цей алгоритм має ряд переваг. Можна регулювати розмірність блоку з яким проводяться обчислення на кожному кроці алгоритму, за рахунок цього може досягатись ефект кешезації обчислень, коли блоки повністю розміщуються у швидкій пам’яті GPU. Також така блочна структура дозволяє працювати з

нерозривними масивами даних на GPU, що зменшує кількість індексних операцій і перевірок, які на графічному прискорювачі є досить затратними. [32]

3.2 Прямі алгоритми UTDU-розвинення СЛАР з матрицями нерегулярної структури

Відомо, що для розв'язування СЛАР

$$Ax = b \tag{3.9}$$

з симетричною додатньо визначеною матрицею найбільш ефективним прямим методом вважається метод Холецького, який здійснює LDL^{T}-розвинення A, де L – нижня трикутна, а D – діагональна матриця. Проте у цьому випадку при побудові паралельних алгоритмів матриця нерегулярної структури розподіляється таким чином, що кожен «хмарочос» міститься цілком в одному процесі. Оскільки кількість та параметри «хмарочосів» наперед невідомі, це може призвести до нерівномірного розподілу даних і порушення балансування процесів. Запропоновано $U^{\mathsf{T}}DU$-розвинення, де U – верхня трикутна, при якому межі розподілу даних проходять не вздовж, а впоперек «хмарочосів».

Отже, знаходження розв'язку системи (3.9) зводиться до розвинення

$$A = U^{\mathsf{T}}DU \tag{3.10}$$

та наступного розв'язування двох трикутних систем

$$U^{\mathsf{T}}y = b, \tag{3.11}$$

$$DUx=y. \tag{3.12}$$

Елементи $U^{\mathsf{T}}DU$-розвинення обчислюються у циклі за n кроків, на кожному з яких виконуються перетворення:

$$u_{ki} = \hat{u}_{ki}/d_k, \quad d_i = \hat{d}_i - \sum_{r=k+1}^{i} u_{ki}\hat{u}_{kr}\ , \quad u_{ij} = \hat{u}_{ji} - \sum_{r=k+1}^{j} u_{kj}\hat{u}_{kr}\ , \tag{2.18}$$

$$i = k + 1, \ldots, n, \qquad j = i + 1, \ldots, n,$$

де u_{ki} та $\hat{u}_{ki}$ − значення елементу матриці U на k і k–1-му кроці, d_i − діагональний елемент матриці D. Перед початком перетворень елементи матриці U покладаються рівними відповідним елементам матриці A, а до D входять діагональні елементи A.

Розв’язування трикутних систем (3.11) та (3.12) виконується відповідно за формулами:

$$y_k = b_k - \sum_{i=1}^{k-1} u_{ik} y_i \ , \quad k = 1, 2, \ldots, n, \tag{3.14}$$

$$x_k = \frac{y_k}{d_k} - \sum_{i=k+1}^{n} u_{ki} x_i \ , \ k = n, n-1 \ , \ldots, 1. \tag{3.15}$$

У випадку розріджених матриць кількість обчислень можна суттєво зменшити, якщо враховувати властивості задачі. З алгоритмічної точки зору це означає, що $U^{\mathsf{T}}DU$-розвинення та розв’язування систем з матрицями нерегулярної структури потребує проведення обчислювальних операцій того ж типу, що й у випадку щільних матриць, але з іншою організацією циклів. Це зумовлено кількома причинами. По-перше, обчислення проводяться лише для тих елементів, які є ненульовими або перетворюються на такі у ході факторизації. По-друге, з огляду на специфіку розрідженості розглядуваних матриць, внутрішні цикли орієнтовані на обробку елементів групами. Окрім того, оскільки розглядаються симетричні матриці, обчислення проводяться лише у верхньому трикутнику.

Позначимо через $NZ(A)$ множину елементів матриці A, які є ненульовими або стають такими у ході обчислень. Тобто, до $NZ(A)$ входять такі a_{ij}, для яких виконується принаймні одна з двох умов:

1) $a_{ij} \neq 0$;

2) для деякого кроку алгоритму з номером k одночасно $a_{kj} \neq 0$ і $a_{ki} \neq 0$.

Тоді розвинення матриці нерегулярної структури на k-му кроці ($k = 1, 2, \ldots, n$) відбувається згідно з ***Алгоритмом 3.1***.

Алгоритм 3.1. $U^{\mathsf{T}}DU$-розвинення розрідженої матриці

Крок 1. Покласти $i = k$.

Крок 2. Якщо $a_{ki} \notin NZ(A)$ перейти на **Крок 7**.

Крок 3. Обчислити $a_{ki} = a_{ki} / a_{kk}$ і покласти $j = i + 1$.

Крок 4. Якщо $a_{ji} \in NZ(A)$, обчислити $a_{ji} = a_{ji} - a_{kj} a_{ki}$.

Крок 5. Якщо $j < k$ покласти $j = j + 1$ і перейти на **Крок 4**.

Крок 6. Обчислити $d_i = d_i - a_{kk}\, a_{ki}$.

Крок 7. Якщо $i < n$ покласти $i = i + 1$ і перейти на **Крок 2**.

Легко бачити, що без знання особливостей множини $NZ(A)$ наведений алгоритм змушує виконувати певну кількість порожніх проходжень циклів, яка може виявитися досить великою, особливо у випадку матриць великих порядків. Цей недолік дозволяє у повній мірі усунути «хмарочосна» схема.

Позначимо через I_k та J_k масиви довжин груп ненульових та нульових елементів відповідно у k-му рядку матриці, K – масив кількостей таких груп у кожному рядку. I_k та J_k введено як частини описаних вище масивів I та J:

$$I = \bigcup_{k=1}^{n} I_k \ , \ J = \bigcup_{k=1}^{n} J_k \ .$$

Для звертання до k-го елемента, наприклад, масиву K, використовуватимемо позначення $K^{(k)}$.

Отже, на кожному k-му ($k = 1, 2, \ldots, n$) кроці $U^{\mathsf{T}}DU$-розвинення матриці в «хмарочосному» форматі виконується за ***Алгоритмом 3.2***.

Алгоритм 3.2. $U^{\mathsf{T}}DU$-розвинення матриці в «хмарочосному» форматі

Крок 1. Покласти $i = 1$.

Крок 2. Покласти $j = 1$.

Крок 3. Покласти $l = 1$ та $t = j + \sum_{r=1}^{i} I_k^{(r)} + \sum_{r=1}^{i-1} J_k^{(r)}$ і обчислити $a_{kt} = a_{kt} / a_{kk}$.

Крок 4. Покласти $m = 1$.

Крок 5. Якщо $a_{st} \notin NZ(A)$, де $s = k + m + \sum_{r=1}^{l} I_k^{(r)} + \sum_{r=1}^{l-1} J_k^{(r)}$, перейти на **Крок 7**.

Крок 6. Якщо $s=t$, то обчислити $d_s=d_s-a_{ks}a_{kt}$, інакше обчислити $a_{st}=a_{st}-a_{ks}\, a_{kt}$.

Крок 7. Якщо $m < I_k^{(l)}$ покласти $m = m + 1$ і перейти на **Крок 5**.

Крок 8. Якщо $l < K^{(k)}$ покласти $l = l + 1$ і перейти на **Крок 4**.

Крок 9. Якщо $j < I_k^{(i)}$ покласти $j = j + 1$ і перейти на **Крок 3**.

Крок 10. Якщо $i < K^{(k)}$ покласти $i = i + 1$ і перейти на **Крок 2**.

Паралельний блочно-циклічний алгоритм U^TDU-розвинення. На основі наведеної схеми розроблено і програмно реалізовано блочно-циклічний алгоритм з паралельною організацією обчислень. Зупинимося детальніше на його особливостях та відмінностях від послідовної схеми.

Розглянемо комп'ютер MIMD-архітектури. Нехай A – квадратна матриця порядку n. Представимо матрицю у блочно-циклічному вигляді наступним чином

$$A = \left(A_1^* \quad A_2^* \quad \ldots \quad A_p^* \right)^T,$$

$$A_i^* = \left(A_j \right)_{j=i+kp}, \quad 0 \le k < n/sp, ,$$

де A_i – блок, що містить s послідовних рядків матриці, p – кількість процесів.

Розташуємо матрицю по процесах таким чином, щоб $A^*_i \in P_i, i = \overline{1, p}$. Не втрачаючи загальності міркувань, припускатимемо, що n / (sp) – ціле число. На кожному кроці з номером m = 0, s, 2s,..., (n – s) процесор з номером $\{m/(sp)\}$, де $\{a\}$ дорівнює остачі від ділення, називатимемо провідним процесом, а блок з номером m/s – провідним блоком. Тоді на m-му кроці U^TDU-розвинення матриці виконується ***Алгоритм 3.3*** (щоб не втрачалося розуміння паралелізму тут не наводяться особливості циклів, пов'язані з «хмарочосною» схемою, вони повністю відтворюють наведені в Алгоритмі 2.2).

Алгоритм 3.3. U^TDU-розвинення матриці (паралельний)

Крок 1. За допомогою операції мультирозсилки массив ненульових елементів u_{ik} ($k = m, m + 1,..., n; i = m, m + 1, ..., \min\{m + s, n\}$) з провідного блоку провіднм процесом розіслати іншим.

Крок 2. Покласти $k = m$.

Крок 3. У кожному процесі перевірити виконання умов додатньої визначеності (d_k>0) або невирожденості ($|d_k| > \text{macheps}\|A\|$, де macheps – значення машинної точності) вихідної матриці A.

Крок 4. У кожному процесі обчислити значення $1/d_k$ та ненульових елементів u_{ki} ($i = k + 1,..., n$), що входять до ($i – m$)-го рядку провідного блоку трикутного розвинення згідно з (2.18):

$$u_{ki} = \hat{u}_{ki}/d_k, \quad i = k + 1, \ldots, n, \qquad u_{ki} \in NZ(U).$$

Крок 5. У кожному процесі обчислити значення ненульових елементів

$$d_i = \hat{d}_i - \sum_{r=k+1}^{i} u_{ki}\hat{u}_{kr}\ , \qquad u_{ij} = \hat{u}_{ji} - \sum_{r=k+1}^{j} u_{kj}\hat{u}_{kr} \qquad u_{ij} \in NZ(U),$$

у провідному блоці ($i = k + 1, k + 2,..., m + s; j = i, i + 1,..., n$) та інших блоках, які зберігаються у відповідному процесі ($i = r, r + 1,..., r + s; j = i, i + 1,..., n; r = m + s(q+ 1), m + s(q+ 1 + p),..., m + s(q+ 1 + n/s - I)$), де q – номер процесу.

Крок 6. Якщо $k < \min\{m + s, n\}$, покласти $k = k + 1$ і перейти на Крок 3, інакше алгоритм завершує роботу.

3.3. Розв'язування трикутних систем

Розв'язування системи (2.16) для кожного $m = 0, s, 2s,..., (n – s)$ виконується за ***Алгоритмом 3.4***, який базується на схемі (3.15).

Алгоритм 3.4. Розв'язування верхньої трикутної системи

Крок 1. У провідному процесі виконати перетворення з m-го по $(m + s)$-ий елемент матриці у рядках з m-го по $(m + s)$-й згідно з (3.15). Тобто, для $k = m, m + 1,..., \min\{I + s, n\}$ обчислити

$$y_k = b_k - \sum_{i=m}^{k-1} u_{ik} y_i\ .$$

Крок 2. За допомогою операції мультирозсилки частину елементів вектора правих частин y_k, де $k = I + 1,..., \min\{m + s, n\}$ з провідного процесу розіслати іншим.

Крок 3. У кожному процесі провести модифікацію елементів правих частин, що знаходяться нижче провідного блоку. Тобто

$$y_k \leftarrow y_k - \sum_{i=m}^{\min\{m+s,n\}} u_{ik} y_i\ ,$$

де $k = r, r + 1, \ldots, r + \min\{r+ s, n\}, \qquad r = m + s(q + 1)$,

$m + s(q + 1 + p),..., m + s(q + 1 + n/s - m))$, де q – номер процесу.

Алгоритм 3.5 для розв'язування системи (3.12) будується згідно з (3.15) за подібним принципом, але з проходженням у зворотньому напрямку (тобто для кожного $m = n – s,\ldots, 2s, s, 0$).

Алгоритм 3.5. Розв'язування нижньої трикутної системи

Крок 1. У провідному процесі виконати перетворення з m-го по $(m + s)$-ий елемент матриці у рядках з m-го по $(m + s)$-й згідно з (3.15).

Тобто, для $k = m + 1,..., \min\{I + s, n\}$ обчислити

$$x_k = \frac{y_k}{d_k} - \sum_{i=m}^{k} u_{ki} x_i .$$

Крок 2. За допомогою операції мультирозсилки частину елементів вектора правих частин x_k, де $k = m + 1,..., \min\{m + s, n\}$ з провідного процесу розіслати іншим.

Крок 3. У кожному процесі провести модифікацію елементів правих частин, що знаходяться вище провідного блоку. Тобто

$$x_k = \frac{y_k}{d_k} - \sum_{i=m-s}^{m} u_{ki} x_i ,$$

де $k = r, r + 1, \ldots, r + s,$ $r = sq, s(q + 1),..., s(q + n/s - I))$, де q – номер процесу.

3.4. Оцінка похибки комп'ютерних розв'язків

Для алгоритмів розв'язування СЛАР з матрицями у «хмарочосному» форматі розроблено обчислювальні схеми знаходження оцінки числа обумовленості матриці, спадкової та обчислювальної похибок. [3]

Для обчислення оцінки числа обумовленості за схемою (3.2)-(3.4) необхідно знайти норми матриці та векторів. Позначимо A_k – масив ненульових значень k-го рядка матриці, яка зберігається у «хмарочосній» схемі. Тобто:

$$A = \bigcup_{k=1}^{n} A_k .$$

Тоді обчислення норми матриці «хмарочосної» структури відбувається згідно з ***Алгоритмом 3.6***.

Алгоритм 3.6. Обчислення норми матриці «хмарочосної» структури

Крок 1. У кожному процесі покласти $g_q = 0$, $l = 0$, $k = 1 + sq$, де q – номер процесу, s – розмір блоку.

Крок 2. Покласти $i = k$.

Крок 3. Покласти $l = l + a_{ki}$.

Крок 4. Якщо $i < |A_k|$, де $|.|$ – кількість ненульових елементів у відповідному рядку, покласти $i = i + 1$ і перейти на **Крок 3.**

Крок 5. Якщо $g_q < l$ то покласти $g_q = l$.

Крок 6. Якщо $k < n$ та $\{k/s\} = 0$, покласти $k = k + 1 + sp$, де p – кількість процесів, і перейти на Крок 2.

Крок 7. Якщо $k < n$, покласти $k = k + 1$ і перейти на **Крок 2.**

Крок 8. За допомогою операцій мультирозсилання та мультизбирання всі процесори обмінюються значеннями g_q і знаходять максимальне серед них :

$$g = \max_{i=1,\dots,p} g_i .$$

Пошук норми вектора, розподіленого блочно-циклічним способом, відбувається згідно з ***Алгоритмом 3.7***.

Алгоритм 3.7. Обчислення норми вектора

Крок 1. У кожному процесі покласти $v_q = z_{1+sq}$, $w_q = y_{1+sq}$, $k = 2 + sq$, де q – номер процесу, s – розмір блоку.

Крок 2. Покласти $v_q = v_q + z_k$, $w_q = w_q + y_k$.

Крок 3. Якщо $k < n$ та $\{k/s\} = 0$, покласти $k = k + 1 + sp$, де p – кількість процесів, і перейти на **Крок 2.**

Крок 4. Якщо $k < n$, покласти $k = k + 1$ і перейти на **Крок 2.**

Крок 5. За допомогою операцій мультирозсилання та мультизбирання всі процесори обмінюються значеннями v_q та w_q і знаходять їхні суми:

$$v = \sum_{i=1}^{p} v_i , w = \sum_{i=1}^{p} w_i .$$

Тоді для системи з матрицею, яка зберігається у «хмарочосному» форматі, знаходження розв’язку з оцінкою числа обумовленості необхідно виконувати за ***Алгоритмом 3.8***.

Алгоритм 3.8. Обчислення оцінки числа обумовленості матриці

Крок 1. За допомогою ***Алгоритму 3.6*** знайти норму матриці g.

Крок 2. За допомогою ***Алгоритму 3.3*** провести розвинення матриці

$$A = U^{\mathsf{T}}DU.$$

Крок 3. За допомогою ***Алгоритму 3.4*** та ***Алгоритму 3.5*** розв'язати СЛАР

$$U^{\mathsf{T}}DUy = e,$$

де компоненти вектора e мають значення 1 або –1, які вибираються таким чином, щоб збільшити норму вектора y.

Крок 4. За допомогою ***Алгоритму 3.4*** та ***Алгоритму 3.5*** розв'язати СЛАР

$$U^{\mathsf{T}}DUz = y.$$

Крок 5. За допомогою ***Алгоритму 3.7*** знайти норми векторів y та z – величини v та w відповідно.

Крок 6. Обчислити оцінку числа обумовленості:

$$c(A) = g\, v / w.$$

Маючи оцінку числа обумовленості та знаючи похибки задання елементів матриці і правих частин, знаходження величини спадкової похибки можна звести до операції [33]

$$e = \frac{c(A)}{1 - c(A)\varepsilon_A}(\varepsilon_A + \varepsilon_b),$$

та виконати на одному процесі.

Для знаходження величини обчислювальної похибки розв'язку СЛАР необхідно виконати множення матриці A, розподіленої за блочно-циклічною схемою, на вектор x. Для зручності вважатимемо, що кожен процес має всі компоненти вектора, які попередньо могли бути зібрані за допомогою операцій мультирозсилки та мультизбирання. Результуючий вектор y, як і матриця, розподілений блочно-циклічним способом. Тоді на кожному k–му кроці ($k = m + l$, $m = 0, s, 2s, \ldots, (n - s)$, $l = qs, (q + p)s, (q + 2p)s, \ldots$, де q – номер процесу, p – кількість процесів, а s – розмір блоку) процесори незалежно проводять обчислення за ***Алгоритмом 3.9***.

Алгоритм 3.9. Множення «хмарочосної» матриці на вектор

Крок 1. Покласти $y_k = 0$.

Крок 2. Покласти $i = 1$.

Крок 3. Покласти $j = 1$.

Крок 4. Покласти $y_k = y_k + x_t\, a_{kt}$, де $t = j + \sum_{r=1}^{i} I_k^{(r)} + \sum_{r=1}^{i-1} J_k^{(r)}$.

Крок 5. Якщо $j < I_k^{(i)}$ покласти $j = j + 1$ і перейти на **Крок 4.**

Крок 6. Якщо $i < K^{(k)}$ покласти $i = i + 1$ і перейти на **Крок 3.**

З використанням наведеного алгоритму знаходження обчислювальної похибки розв'язку x для системи з матрицею A (яка була розвинута у добуток $U^{\mathsf{T}}DU$) та правою частиною b, де наведені матриці та вектори розподілені згідно з блочно-циклічною схемою, виконується за ***Алгоритмом 3.10***.

Алгоритм 3.10. Знаходження обчислювальної похибки

Крок 1. За допомогою ***Алгоритму 3.9*** обчислити добуток

$$r = Ax.$$

Крок 2. У кожному процесорі обчислити всі значення

$$r_k = b_k - r_k,$$

де $k = m + l$, $m = 0, 1, \ldots, s - 1$, $l = qs, (q + p)s, (q + 2p)s, \ldots, q$ – номер процесу, p – кількість процесів, а s – розмір блоку.

Крок 3.За допомогою ***Алгоритму 3.4*** та ***Алгоритму 3.5*** розв'язати СЛАР

$$U^{\mathsf{T}}DU\delta = r.$$

Крок 4. У кожному процесорі обчислити всі значення

$$r_k = x_k - \delta_k,$$

де $k = m + l$, $m = 0, 1, \ldots, s - 1$, $l = qs, (q + p)s, (q + 2p)s, \ldots$.

Крок 5. За допомогою ***Алгоритму 3.7*** знайти норми векторів δ та r – величини v та w відповідно.

Крок 6. Обчислити значення обчислювальної похибки за формулою

$$e = v / w.$$

3.5. Прискорення та ефективність паралельних алгоритмів для матриць нерегулярної структури

Для обчислення коефіцієнтів прискорення та ефективності алгоритму U^TDU-розвинення скористаємося формулами:

$$S_p=T_1 / T_p, E_p=S / p, \qquad (3.16)$$

де p – кількість процесів, T_1 – час розв'язування задачі на одному процесі, T_p – час розв'язування на p процесах. Часи виконання обчислюватимемо як

$$T_1 = Nt, \quad T_p = Nt + Mt_o + Qt_c, \qquad (3.17)$$

де N – кількість арифметичних операцій (додавання та множення), t – час виконання однієї арифметичної операції, M – кількість обмінів, t_o – час виконання одного обміну, Q – кількість синхронізацій між процесами, t_c – час однієї синхронізації.

Позначимо через η_i кількість ненульових елементів у i-му рядку верхньої трикутної матриці U^TDU-розвинення розрідженої симетричної матриці. Тоді справедлива теорема, аналогічна теоремі 3.1 для LDL^T-розвинення.

Теорема 3.3. Для U^TDU-розвинення розрідженої симетричної матриці потрібно виконати

$$N_1 = \sum_{i=1}^{n} \eta_i^2$$

арифметичних операцій з плаваючою комою.

Доведення. На i-му (i = 1, 2, ..., n) кроці для модифікації провідного рядка згідно (2.18) необхідно виконати одне ділення і η_i–1 множень, а для перетворення елементів k-ого з η_i – 1 рядків, номери яких відповідають індексам ненульових елементів провідного рядка, за η_i–k множень і додавань, де k = 1, 2, ..., η_i –1.

Таким чином, загальна кількість арифметичних операцій з плаваючою комою на i-му кроці становить $\eta_i + (\eta_i - 1)\,\eta_i = \eta_i^2$. Звідси маємо $N_1 = \sum_{i=1}^{n} \eta_i^2$.

Далі для визначення T_p з (3.16) необхідно оцінити N_p, M_p, Q_p з (3.17). В ідеальному випадку $N_p = N_1/p$. Проте, це можливо при повній збалансованості завантаження процесів на кожному кроці алгоритму. Наприклад, у наведених вище алгоритмах частина обчислень виконується без розпаралелювання: або тільки одним процесом, або всі процеси виконують одні й ті ж обчислення з одними і тими ж даними.

Теорема 3.4. Одновимірний блочно-циклічний алгоритм $U^{\mathrm{T}}DU$-розвинення розрідженої матриці вимагає виконання не менше

$$N_p^{(0)} = \frac{1}{p}\left(\sum_{i=1}^{n} \eta_i^2 + (p-1)\sum_{i=1}^{n} \eta_i \right)$$

арифметичних операцій з плаваючою комою, тобто $N_p \geq N_p^{(0)}$.

У припущенні повної збалансованості завантаження процесів на кожному кроці алгоритму кожний процес виконує не менше

$$(N_1 + (p-1)N^{(+)})/p \geq N_p^{(0)}$$

арифметичних операцій з плаваючою комою. У загальному випадку величина N_1/p замінюється сумою найбільших кількостей операцій, виконуваних одним процесом на кожному кроці циклу по *i*. Таким чином, $N_p \geq (N_1 + (p-1)N^{(+)})/p \geq N_p^{(0)}$.

Теорема 3.5. Якщо сітка процесів базується на топології «гіперкуб», то кількість обмінів (синхронізацій) між процесами для одновимірного блочно-циклічного паралельного алгоритму $U^{\mathrm{T}}DU$-розвинення розрідженої матриці оцінюється величиною

$$Q_p \approx \frac{n}{s}\log_2 p,$$

а загальна кількість даних, якими обмінюються процеси, становить

$$M_p \approx \log_2 p \sum_{i=1}^{n} \eta_i$$

двійкових слів.

На кожному кроці циклу по *i* виконується по одній операції мультирозсилки масиву з $\sum_{i=m+1}^{\min\{m+s,n\}} \eta_i$ ненульових елементів провідного блоку. Якщо для цієї операції використовується алгоритм дерева, то при одній мультирозсилці виконується приблизно $\log_2 p$ синхронізацій, а процеси обмінюються приблизно $\log_2 p \sum_{i=m+1}^{\min\{m+s,n\}} \eta_i$ двійковими словами. Отже, загальна кількість синхронізацій оцінюється величиною

$$Q_p \approx n/s \log_2 p,$$

при цьому загальна кількість даних, якими обмінюються процеси, становить приблизно $M_p \approx \log_2 p \sum_{i=1}^{n} \eta_i$.

Теорема 3.6. Для паралельного блочно-циклічного алгоритму $U^{\mathrm{T}}DU$-розвинення мають місце наступні оцінки прискорення та ефективності для комунікаційної мережі типу гіперкуб:

$$S_p \le p\left(1+\frac{p-1}{N_1}\sum_{i=1}^{n}\eta_i+\frac{p\log_2 p}{N_1}\tau_{1s}\right)^{-1}, \quad E_p \le 1-\left(\frac{p-1}{N_1}\sum_{i=1}^{n}\eta_i+\frac{p\log_2 p}{N_1}\tau_{1s}\right),$$

де $N_1=\sum_{i=1}^{n}\eta_i^2$, $\tau_{1s}=\frac{t_c}{t}\frac{n}{s}+\frac{t_o}{t}\sum_{i=1}^{n}\eta_i$, t – час виконання однієї арифметичної операції, t_o – час одного обміну, а t_c – час, необхідний для однієї синхронізації процесів.

Отже, враховуючи також результати теореми 3.3 і співвідношення (3.16), отримаємо шукані оцінки для коефіцієнтів прискорення та ефективності одновимірного блочного циклічного паралельного алгоритму $U^{\mathrm{T}}DU$-розвинення розрідженої симетричної матриці.

Зазначимо, що наведені оцінки досяжні у випадку розподілу між процесами однакової кількості ненульових елементів.

З отриманих оцінок випливає висока масштабованість паралельного алгоритму. При цьому частка комунікаційних витрат у загальному часі розв’язування задачі тим менша, чим більший порядок системи.

Висновки до розділу

Як можна побачити існує багато методів вирішення систем лінійних алгебраїчних рівнянь з розрідженими матрицями. Але при паралелізації цих методів виникає загальна задача розподілення та розбиття задачі на декілька частин та процесорних ядер. Зазвичай для цього проводиться аналіз задачі та її структури (а саме кількості та щільності ненульових елементів матриці). Для автоматизації процесу аналізу структури розрідженої матриці та її ненульових елементів і була розроблена нейромережа багаторівневої структури.

4 РОЗРОБКА НЕЙРОМЕРЕЖІ ДЛЯ АНАЛІЗУ СТРУКТУРИ РОЗРІДЖЕНИХ МАТРИЦЬ

Оскільки основною задачею нейронної мережі буде аналіз зображень, то був обраний найбільш пристосований тип нейромережі, а саме згорткова. В результаті була розроблена нейромережа наступної структури зображеної на рис 4.1

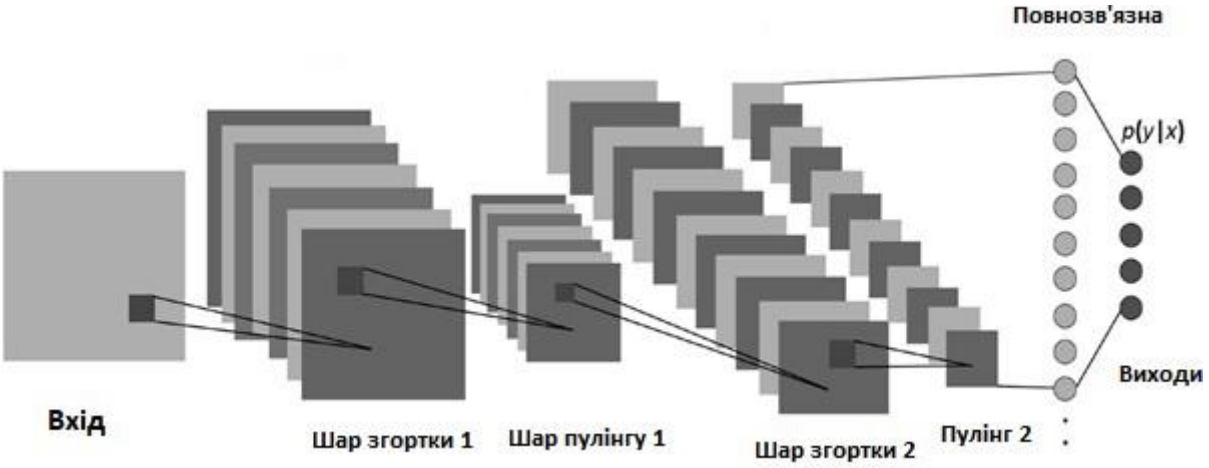


Рисунок 4.1 – Структура згорткової нейромережі.

Згорткова нейронна мережа складається з декількох шарів. Ці шари можуть бути трьох типів:

Згортковий (Convolutional): Згоркові шари складаються з прямокутної сітки нейронів. Він потребує, щоб попередній шар також був прямокутною сіткою нейронів. Кожен нейрон приймає входи з прямокутного розділу попереднього шару; ваги для цього прямокутного розділу однакові для кожного нейрона в згортковому шарі. Таким чином, згортковий шар - це лише зображена згортка попереднього шару, де вага визначає фільтр згортки. Крім того, у кожному згортковому шарі може бути кілька сіток; кожна сітка приймає вхідні дані всіх сіток попереднього шару, використовуючи потенційно різні фільтри.

Пулінг (Max-Pooling): після кожного згорткового шару, може існувати шар пулінгу. Пулінговий (агрегувальний) шар приймає невеликі прямокутні блоки з згорткового шару і зменшує його, щоб виготовити єдиний вихід з цього блоку. Існує декілька способів зробити це об'єднання, наприклад, прийняття середнього або максимального, або вивчена лінійна комбінація нейронів у блоці. В данному випадку використовуються шаои макс-пуллінгу. Тобто вони беруть максимум блоку, який вони об'єднують.

Повноз'єднаний (fully-conected): після декількох згорткових і макс-пулінгових шарів, класифікація високого рівня в нейронній мережі виконуються через повноз'єднанні шари. Повноз'єднаний шар приймає всі нейрони на попередньому шарі (будь то повноз'єднаний, агрегувальний або згортковий) і підключається до кожного окремого нейрона, який він має. Повноз'єднаннї шари більше не розміщені в просторовій структурі тому після повноз'єднанного шару не може бути шарів згортки.

Розроблена нейромережа складається з наступних компонентів або шарів які використовуються декілька разів.

4.1 Шар згортки (convolution)

У згорткової нейронної мережі в операції згортки використовується лише обмежена матриця ваг невеликого розміру, яку «рухають» по всьому оброблюваного шару (на самому початку - безпосередньо по вхідному зображенню), формуючи після кожного зсуву сигнал активації для нейрона наступного шару з аналогічною позицією. Тобто для різних нейронів вихідного шару використовуються одна і та ж матриця ваг, яку також називають ядром згортки ... Тоді наступний шар, що вийшов в результаті операції згортки такою матрицею ваг, показує наявність даної ознаки в оброблюваному шарі і її координати, формуючи так звану карту ознак (англ. feature map).

Операція згортки відбувається за наступною формулою:

$$x_{ij}^{l} = \sum_{a=-\infty}^{+\infty} \sum_{b=-\infty}^{+\infty} w_{ab}^{l} \cdot y_{(i \cdot s-a)(j \cdot s-b)}^{l-1} \quad \forall i \in (0, \ldots, N) \quad \forall j \in (0, \ldots, M) \tag{4.1}$$

де підрядкові індекси i, j, a, b — це індекси елементів матриці, а s — величина шагу згортки (stride).

Надрякові l и l−1 — це індекси шарів нейромережі.

xl-1 — вихід попереднього шару, або зображення yl-1 — це xl-1 після проходження функції активації wl — ядро згортки

xl — результат операції згортки. Операція згортки проходить окремо по кожному елементу ij матрицы xl, розмірність якої (N,M)(N,M).

При зворотньому методі (backpropagation):

Давайте припустимо, що у нас є деяка функція помилки, E, і ми знаємо значення помилки на нашому згортковому шарі.

Помилка, яку ми знаємо, і що нам потрібно обчислити для попереднього шару, є частковою величиною E по відношенню до кожного виходу нейрона.

Необхідно з'ясувати, який градієнтний компонент для кожного ваги, застосовуючи правило ланцюга. В правилі ланцюга ми повинні сумувати внесок усіх виразів, в яких відбувається зміна.

$$\frac{\partial E}{\partial \omega_{ab}} = \sum_{i=0}^{N-m} \sum_{j=0}^{N-m} \frac{\partial E}{\partial x_{ij}^{\ell}} \frac{\partial x_{ij}^{\ell}}{\partial \omega_{ab}} = \sum_{i=0}^{N-m} \sum_{j=0}^{N-m} \frac{\partial E}{\partial x_{ij}^{\ell}} y_{(i+a)(j+b)}^{\ell-1} \tag{4.2}$$

де l – l тий шар, де l = 1 - перший шар, а l = L - останній шар, x має розмірність H × W і має i j як ітератори,

фільтр або ядро ω має розмірність $k1 \times k2$ має m та n як ітератори

ωab, - вагова матриця, яка з'єднує нейрони з шаре l з нейронами шару l-1.

$x^l{}_{ij}$ це зведений вхідний вектор на шарі l, а також зміщення, представлене як

$$x_{ij}^{l} = \sum_a \sum_b \omega_{ab}^{l} o_{(i+a)(j+b)}^{l-1} + b^{l}$$

4.2 Шар агрегування (pooling)

Операція агрегування слугує поступовому скороченню просторового розміру представлення для зменшення кількості параметрів та об'єму обчислень у мережі, і відтак також для контролю перенавчання.

Шар пулінгу бере деякий блок k × k і на вихід подає одне значення, яке є максимальним у цьому блоці. Наприклад, якщо на вхід подається шар N × N, вони потім виведуть шар $\frac{N}{k} \times \frac{N}{k}$, так як кожен k × k блок зводиться до лише одного значення за допомогою max функції.

Використовується метод максимального агрегування (maxpooling) із фільтрами розміру 2×2, що застосовуються з кроком 2.

Макс-агрегування має багато переваг:

- зменшує розмір карти рис та кількість параметрів, що дозволяє запобігти перетренуванню;
- зменшує функціональні карти, зберігаючи найважливіші риси;
- робить мережу інваріантною для малих перетворень, спотворень та помилок на вхідному зображенні (невелике спотворення на вході не змінить висновок шару пулінгу - оскільки ми беремо максимальне значення найближчих елементів).

4.3 Повноз'єднаний шар (fully-conected)

Після шарів згортки ми отримаємо безліч карт ознак. Їх з'єднаємо в один вектор і цей вектор подамо на вхід fully connected мережі.

Формула для fc-слоя (fully connected) виглядає так:

$$x_i^l = \sum_{k=0}^{m} w_{ki}^l y_k^{l-1} + b_i^l \qquad \forall i \in (0, \dots, n) \tag{4.3}$$

4.4 Функція втрат

Завершальний етап мережі - функція, яка оцінює якість роботи всієї моделі (2.12). Функція втрат знаходиться в самому кінці, після всіх шарів мережі.

$$E = \sum_{i=0}^{n} \frac{1}{2}(y_i^{truth} - y_i^l)^2 \tag{4.4}$$

де n — кількість класів, y^l — вивід моделі, а y^{truth} — правильні відповіді.

4.5 Відкидання (Dropout)

Відкидання - методика регуляризації для глибоких нейронних мереж. Навіть найсучасніші моделі, які мають точність 95%, збільшують точність на 2%, лише додаючт відкидання, що є досить значним виграшем на цьому рівні.

Функція Dropout використовується для запобігання перетренування (overfitting) при дуже простій ідеї. Під час навчання, при кожній ітерації нейрон тимчасово "скидається" або вимикається з ймовірністю p. Це означає, що всі входи

та виходи цього нейрона будуть вимикатися під час поточної ітерації. Відкинуті нейрони відновлюються з ймовірністю p на кожному етапі тренування, тому відключений нейрон на одному кроці може бути активним у наступному. Гіперпараметр p називається швидкістю відсіву, і це зазвичай становить приблизно 0,5, що відповідає 50% викидних нейронів.

Хоча ми і навмисно вимикаємо нейрони мережа стає працювати краще. Причиною цього є те, що відключення заважає мережі занадто залежати від невеликої кількості нейронів і змушує кожен нейрон працювати самостійно.

Відкидання можна застосувати до вузлів вхідного шару або прихованого шару, але не до вихідних вузлів. Краї з викинутих вузлів вимкнено.Вузли, які були обранні для відключення, змінюються на кожному навчальному етапі. Також відкидання не застосовується під час тестування після тренування мережі, це робиться лише в процесі навчання.

4.6 Тренування нейромережі

Для тренування мережі був застосований метод зворотнього поширення помилки (backpropagation). Алгоритм навчання можна розділити на дві фази: розповсюдження та оновлення ваги.

Етап 1: розповсюдження

Кожне поширення передбачає наступні кроки:

Крок 1. Розповсюдження вперед через мережу для генерації вихідних значень

Крок 2. Розрахунок вартості (термін помилки)

Крок 3. Розповсюдження вихідних активацій назад через мережу, використовуючи ціль тренувального шаблону, для генерації дельти (різниця між цільовими та фактичними вихідними значеннями) всіх вихідних і прихованих нейронів.

Етап 2: оновлення ваги

Для кожного ваги слід дотримуватися наступних кроків:

Крок 1. Дельта виведення ваги та активація введення множиться, щоб знайти градієнт ваги.

Крок 2. Співвідношення (відсоток) градієнта ваги віднімається від ваги.

Це співвідношення (відсоток) впливає на швидкість і якість навчання; це називається швидкістю навчання. Чим більше співвідношення, тим швидше потягує нейрон, але чим менше співвідношення, тим точніше тренування. Знак градієнта ваги вказує на те, чи помилка залежить від ваги безпосередньо або навпаки. Тому вага повинна бути оновлена в зворотному напрямку, "спускаючись" на градієнт.

Навчання повторюється (на нових партіях), поки мережа не виконає належну роботу.

У вигляді псевдокоду алгоритм можна подати наступним чином:

```
ініціалізувати ваги мережі (часто малі випадкові числа)
do
    forEach тренувальні приклади як ex
        передбачення = результат-нейромережі(network, ex) // прямий хід
        дійсний = результат-прикладу(ex)
        обчислити помилку (передбачення - дійсний) на виходах мережі
        обчислити ΔΔ$w_h$ для всіх ваг від внутрішнього до вихідного шару // зворотній хід
        обчилити Δ$w_i$ для всіх ваг від вхідного до внутрішнього шару // продовження звторнього ходу
        оновити ваги мережі // виключаючи вхідний шар
until усі приклади розпізнано коректно або не досягнутий інший критерій зупинки
return the network
```

Висновки до розділу

В цьому розділі був сформульований формальний опис алгоритмів розв’язання задачі та методів побудови і структури використанної нейромережі. В нейромережі були виділенні основні шари та схему використану при їх побудові.

5 ОПИС ПРОГРАМНОГО ТА ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

5.1 Засоби розробки

Розробка проекту проводиться за допомогою мови програмування Python. Дана мова буде зручна як у реалізації алгоритмів так і при створенні частини нейромережі. Інтерфейс та взаємодія користувачів забезпечується через консольний термінал. Реалізація проекту використовуе такі технології, як TensorFlow, CUDA, Keras та інші.

Далі буде розглянуто переваги використаних технологій та мов програмування.

Python. Широко використовувана інтерпретована, динамічна, мова програмування високого рівня. Його філософія дизайну підкреслює читабельність коду, а його синтаксис дозволяє програмістам, висловити поняття в меншій кількості рядків коду, ніж це було б можливо в інших мовах програмування.

Python підтримує кілька парадигм програмування, в тому числі об'єктноорієнтований, імперативний, функціональний та процедурний підхід до програмування. Він має динамічну систему типів, автоматичне керування пам'яттю, те широку стандартну бібліотеку. Крім цього спільнотою відкритого вихідного коду, розробляються велика кількість допоміжних бібліотек для цієї мови програмування. Однією з таких бібліотек є TensorFlow за допомогою якої реалізована побудова нейромережі.

TensorFlow - це бібліотека програмного забезпечення з відкритим кодом для численних обчислень з високою ефективністю. ЇЇ гнучка архітектура дозволяє легко розгортати обчислення на різних платформах (процесори, графічні процесори, TPU), а також від настільних комп'ютерів до кластерів серверів для мобільних пристроїв. Спочатку розроблена дослідниками та інженерами команди Google Brain в організації AI Google, вона має сильну підтримку для машинного навчання та глибокого навчання, а гнучкі числові обчислення використовуються в багатьох інших наукових областях.

CUDA - це модель паралельної обчислювальної платформи та інтерфейсу прикладного програмування (API), розроблена компанією Nvidia. Це дозволяє розробникам програмного забезпечення та розробникам програмного забезпечення використовувати графічний процесор (GPU) з підтримкою CUDA для обробки загального призначення - підхід, який називається GPGPU (загальнооб'єдноване обчислення на блоках графічної обробки). Платформа CUDA - це програмний рівень, який забезпечує прямий доступ до набору віртуальних наборів графічних процесорів та паралельних обчислювальних елементів для виконання обчислювальних ядер.

Платформа CUDA призначена для роботи з мовами програмування, такими як C, C ++ і Fortran. Ця доступність спрощує паралельне програмування фахівцями використовувати ресурси GPU, на відміну від попередніх APIінтерфейсів, таких як Direct3D та OpenGL, які вимагають високих навичок графічного програмування. Крім того, CUDA підтримує схеми програмування, такі як OpenACC та OpenCL. Коли він вперше був представлений компанією Nvidia, назва CUDA була абревіатурою Compute Unified Device Architecture, але Nvidia згодом відмовилась від використання акроніму.

Keras - бібліотека нейронних мереж із відкритим кодом, написана на Python. Вона здатна працювати з TensorFlow, Microsoft Cognitive Toolkit, Theano або MXNet. Розроблена для швидкого експерименту з глибокими нейронними мережами, вона зосереджена на тому, щоб бути зручною, модульною та розширюваною. Вона була розроблена як частина дослідницької роботи проекту ONEiros (Open-ended Neuro-Electronic Intelligent Robot Operating System), і її основним автором і супроводжувачем є Франсуа Шолле, інженер Google.

У 2017 році команда Google TensorFlow вирішила підтримати Keras у базовій бібліотеці TensorFlow.Keras задумано як інтерфейс, а не самостійний механізм навчання. Він пропонує більш високий рівень інтуїтивно зрозумілого набору абстракцій, що полегшує розробку глибоких моделей навчання незалежно від використовуваного обчислювального бекенда.

5.2 Вимоги до техніного забезпечення

Представлений програмний продукт являє собою программу, для роботи якої, необхідна робоча странція з наступним технічним забезпеченням:

а) технічне забезпечення:

1) процесор з тактовою частотою не нижче 2 ГГц;
2) об'єм оперативної пам’яті не менше 2 Гб;
3) ПЗУ типу SSD або HDD об’ємом не менше 20 ГБ;

б) програмне забезпечення:

1) операційна система – Windows версії не нижче Windows 7;
2) інтерпретатор Python не нижче версії 3;
3) Наступні бібліотеки для Python : TensorFlow, scipy, pyplotlib, Keras, pillow.

Для для тренування нейромережі потрібне наступне програмне та технічне забезпечення:

а) комп'ютер з характеристиками не нижче:

1) процесор з тактовою частотою 2.5 ГГц;
2) оперативна пам’ять 4 Гб;
3) ПЗУ типу SSD або HDD об’ємом не менше 20 ГБ;
4) Відео прискорювач Nvidia з підтримкою технології CUDA (GeForce GTX 590 та краще) .

б) програмне забезпечення:

1) операційна система – Windows версії не нижче Windows 7;
2) інтерпретатор Python не нижче версії 3;
3) Наступні бібліотеки для Python : TensorFlow, scipy, pyplotlib, Keras, pillow;
4) Програмний пакет CUDA.

в) комп’ютерна периферія, до складу якої входить:

1) монітор;
2) мишка;
3) клавіатура.

5.3 Архітектура програмного забезпечення

5.3.1 Опис процесу діяльності

Розглянемо дії які періодично буде виконувати система при виборі користувачем введення даних, за допомогою UML діаграми діяльності (рис. 5.1).

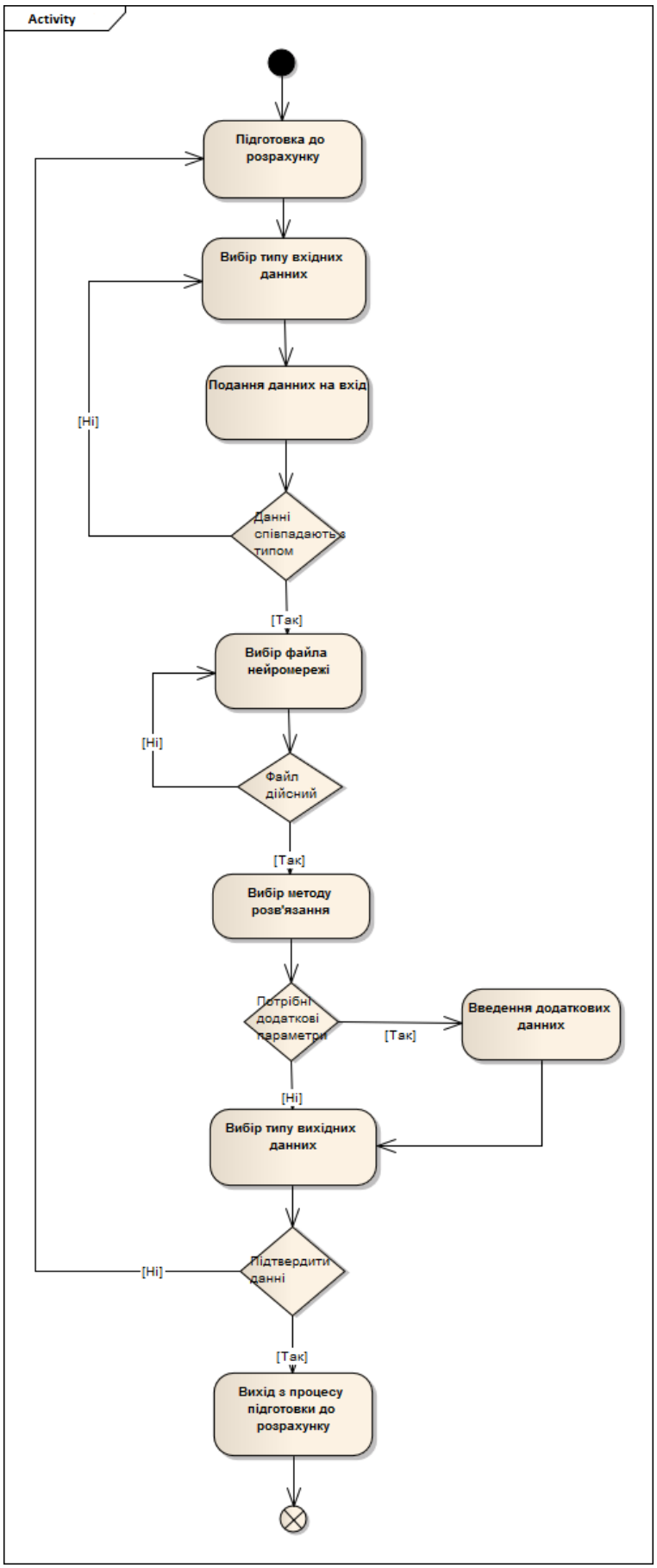


Рисунок 5.1 – Схема структурна діяльності. Процес введення даних для обробки

Детальний опис схеми структурної діяльності процесу внесення початкових даних наведено у таблиці 5.1.

Таблиця 5.1 – опис діяльності процесу внесення даних

Варіант діяльності	**Опис варіанту діяльності**
Підготовка до розрахунку	Загальна підготовка до введення даних (Головне меню)
Вибір типу вхідних даних	Обирається тип даних, який буде подаватися на вхід системи
Подання даних на вхід	Дані подаються на вхід до системи
Данні співпадають з типом	Перевірка на співпадіння даних які поданих на вхід із зарані обраним типом. Якщо, тип не співпадає – то повернення до вибору типу вхідних даних.
Вибір файла нейромережі	Файл нейромережі подається на вхід до системи
Файл дійсний	Перевірка на дійсність файлу поданного на вхід. Якщо, обраний файл не є дійсним – то повернення до вибору файлу нейромережі
Вибір методу розв’язання	Обирається метод, який буде використаний при подальшому розрахунку
Потрібні додаткові параметри	Обрання необхідності додання додаткових параметрів до даних наданих системі. Якщо, відповідь затверджена – то перехід до введення додаткових параметрів.
Введення додаткових параметрів	Додаткові параметри подаються на вхід до системи
Вибір типу вихідних даних	Обирається тип даних, який буде отриманий на виході із системи
Підтвердити дані	Перевірка на дійсність та коректність усієї поданої інформації. Якщо, інформація та дані незадовільні то перехід до підготовки до розрахунку

Продовження таблиці 5.1.

Вихід з процесу підготовки до розрахунку	Перехід до процесу аналізу та роботи с обраними даними.

5.3.2 Схема структурна класів

На рисунку 5.2 представлена структурна схема класів, які відповідають за виконання таких функцій програми, відображення інтерфейсу користувача, процесинг даних, їх аналіз і тд.

Діаграма містить 9 класів, а саме:

- «Data» – клас що виступає в ролі класу, для збереження даних та передачі їх від вводу користувача до інших класів.;
- «UI» – клас який представляє клас взаємодії з користувачем. А, саме прийняття введених даних, передача їх до класу «Data», відображення інформації, тощо;
- «Procesor» – клас який конвертує та перетворює інформацію до належного вигляду та інтерпретує результат аналізу задля подання її в клас «Solver»;
- «Solver» – клас який на підставі даних отриманих з класу «Procesor» займається вирішення систем рівнянь;
- «Model» – основний клас секції аналізу, працює з даними отриманими на вході та ініціалізує аналіз моделі;
- «Layer» – клас відповідальний за головну сторінку, сторінку розширеного перегляду та додання нових роздач;
- «Rating_count» – базовий клас усіх шарів нейромережі, який містить загальні методи;
- «Conv2D» – клас який представляє собою шар згортки. Є підкласом класу «Layer»;
- «MaxPooling» – клас який представляє собою шар агрегації (пулінгу). Є підкласом класу «Layer»;

– «Dense» – клас який представляє собою повноз’єднаний шар (fully-conected). Є підкласом класу «Layer»;

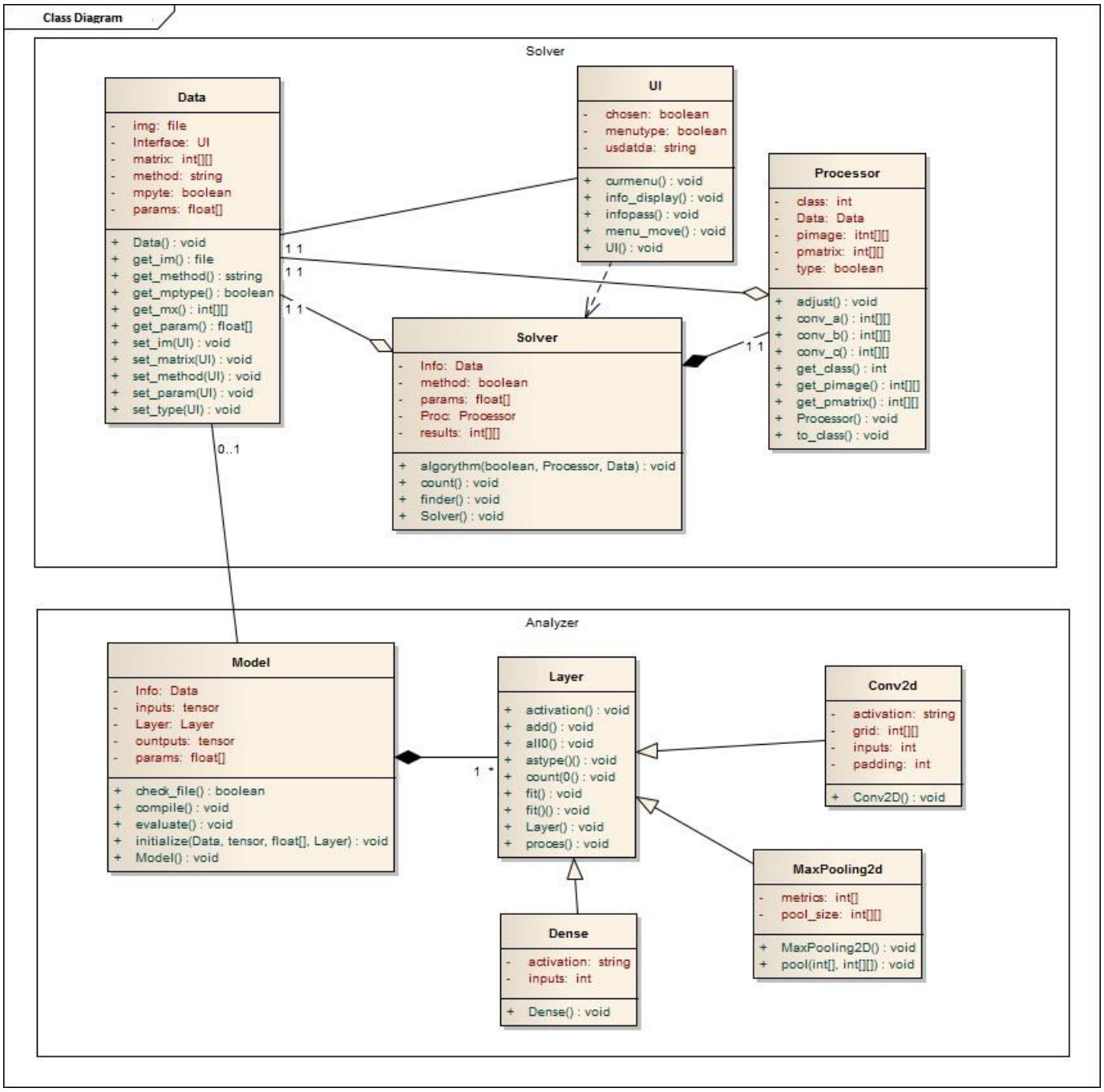


Рисунок 5.2 – Схема структурна класів

5.3.3 Схема структуна послідовності

На рисунку 5.3 та 5.4 представлені схеми структурної послідовності. На даних семах представлені послідовності дій при передачі даних системою (рис.5.3) та при доданні користувачем інформації про файл нейромережі та ін . (рис 5.4). Також визначені актори та їх дії, що необхідні

для виконання поставлених задач.

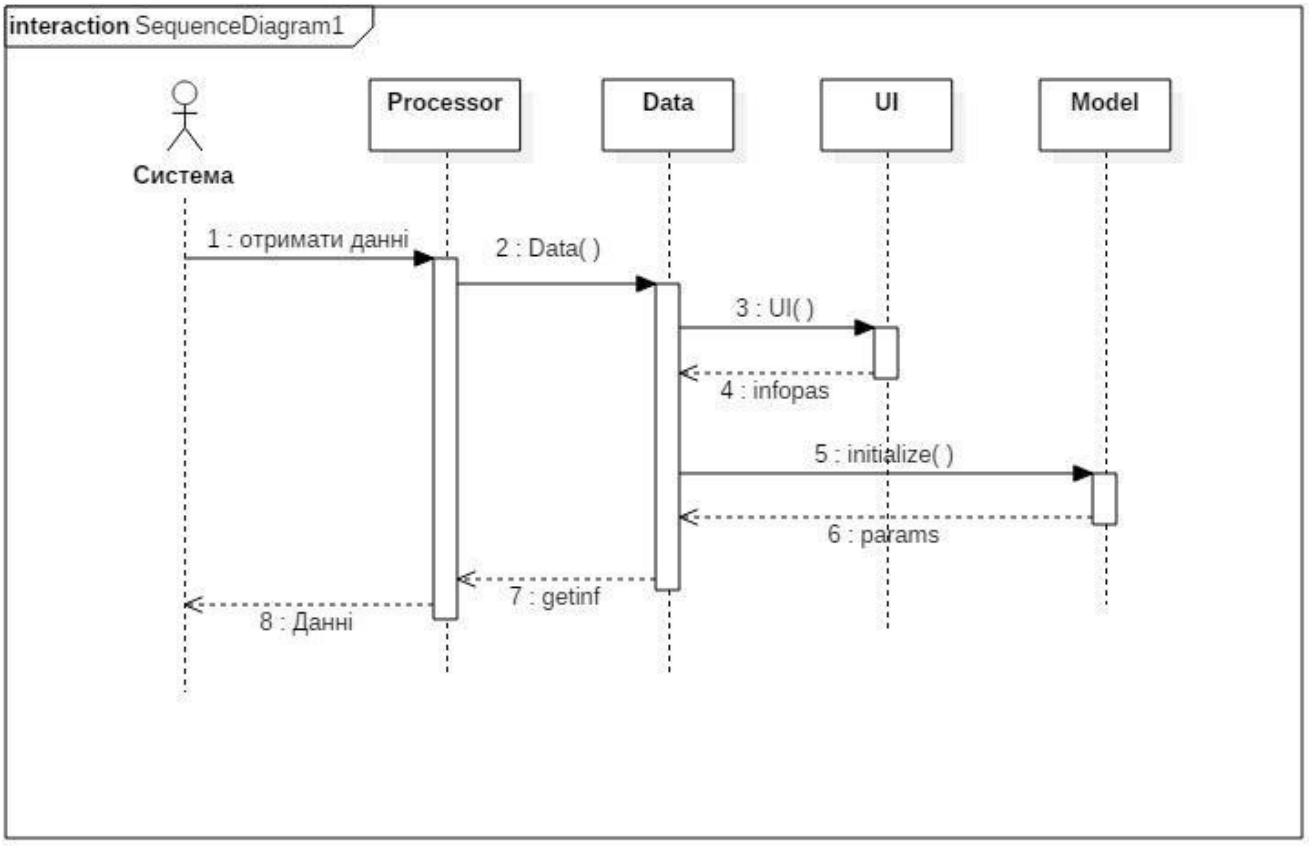


Рисунок 5.3 – Схема структурна послідовності передачі даних

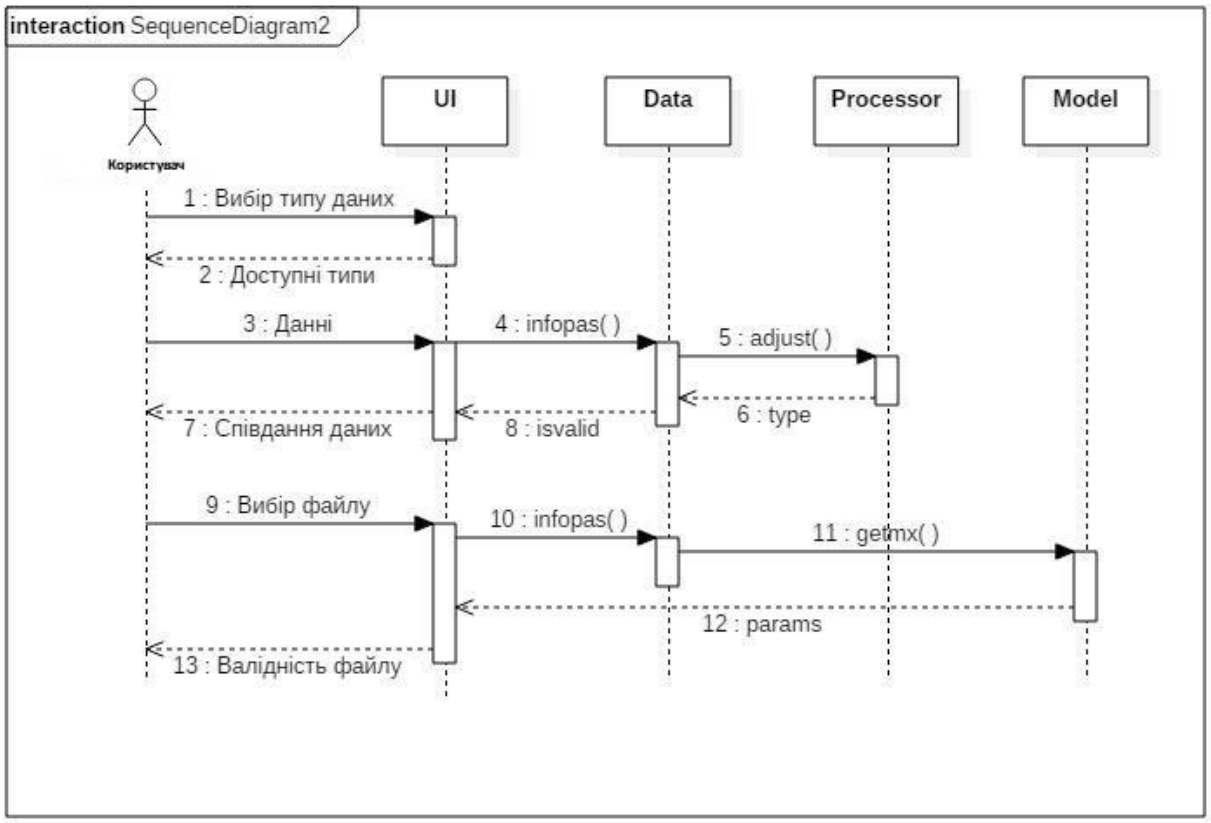


Рисунок 5.4 – Схема структурна послідовності додання даних користувачем

5.3.4 Схема структурна компонентів

На рисунку 5.5 представлена схема структурна компонентів, на якій зображено компоненти системи, що необхідні для функціонування ресурсу, та взаємозв'язки між ними.

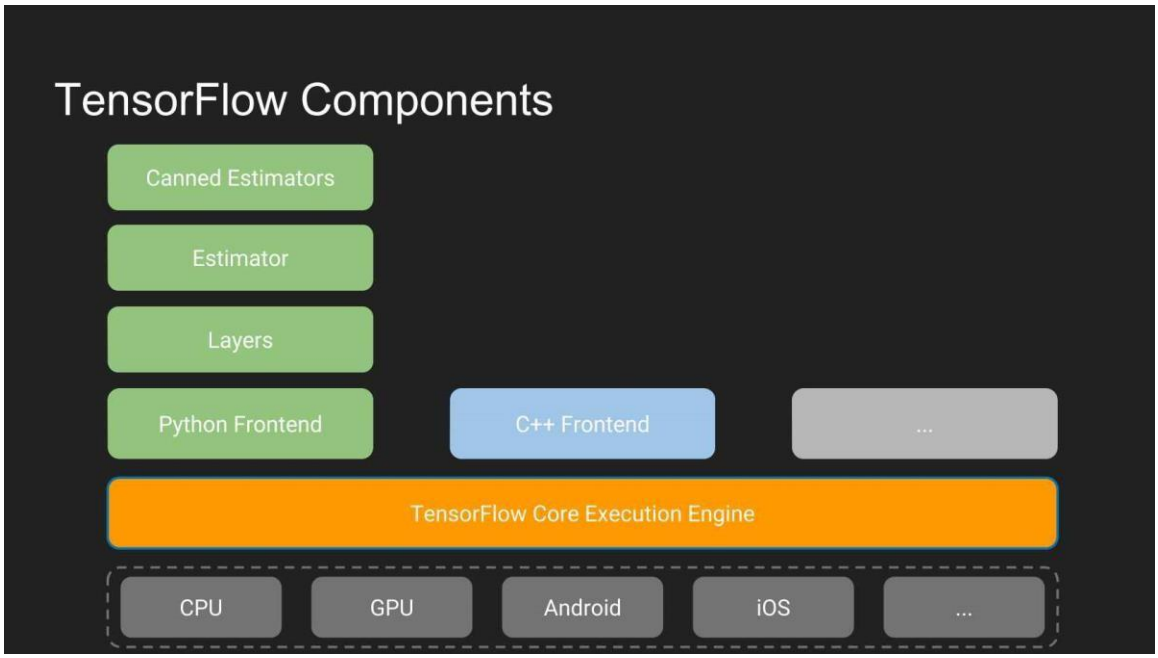


Рисунок 5.5 – Схема структурна компонентів

Висновки до розділу

Розглянуто основні особливості розробки програмних застосувань із нейромережами на мові програмування Python з використанням бібліотеки TensorFlow. До системи, що розробляється були поставлені мінімальні технічні вимоги для функціонування системи. Розроблена схема архітектури програмного забезпечення повністю відповідає поставленим вимогам. Під час розробки було сформовано та розроблено ієрархію класів, які відповідають поставленим вимогам та забезпечують необхідний функціонал системи.

Описані особливості розробки системи, зображені діаграма класів, архітектура програмного забезпечення та мережа взаємодії системи.

6 ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ ВИПРОБУВАНЬ

Розглянемо приклади з СЛАР, отриманими при математичному моделюванні з використанням програмного комплексу ЛІРА – кластер.

6.1 Статичний розрахунок напруго-деформованого стану промислової споруди.

Загальний вид конструкції та використовувана скінченно-елементна сітка подані на рис. 6.1.

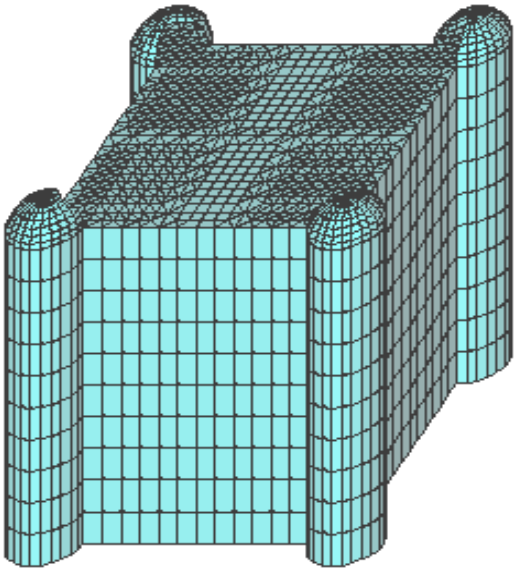


Рисунок 6.1 – Загальний вигляд конструкції

Конструкцію розбито на 13876 скінченних елементів. У результаті сформовано скінченно-елементну сітку з 7563 вузлами. Після дискретизації отримано СЛАР з матрицею порядку 44436. Матриця цієї системи до застосування алгоритму впорядкування має стрічкову структуру (рис.6.2) з щільністю (заповненістю) 21%.

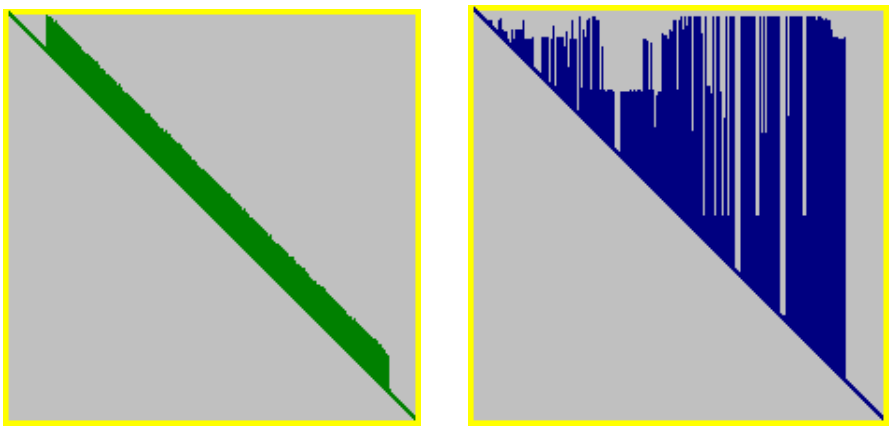


Рисунок 6.2 – Портрет матриці до та після впорядкування елементів

Розрахований розподіл напруги для одного з випадків завантаження зображено на рис. 6.3.

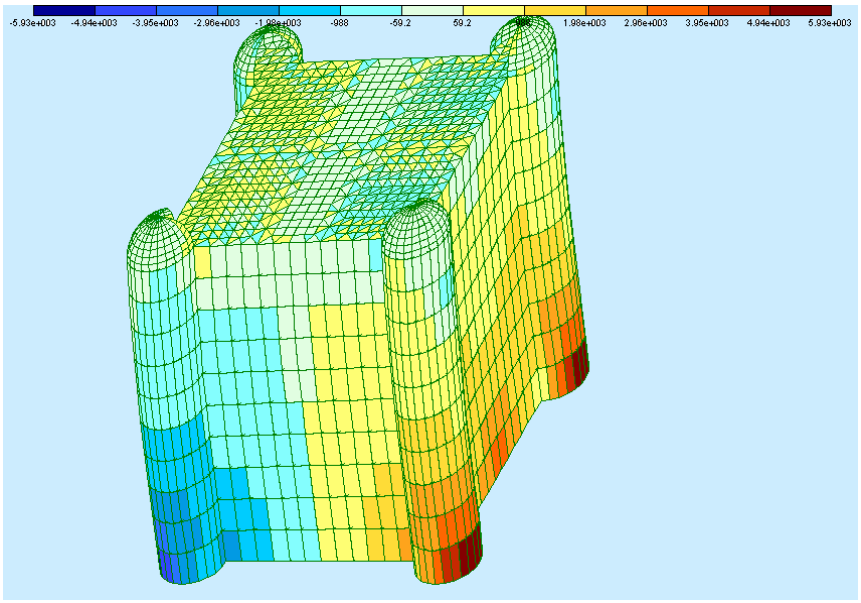


Рисунок 6.3 – Розрахований розподіл напруги

Застосування алгоритму мінімальної степені для зміни впорядкування ненульових елементів матриці дозволила зменшити заповненість до 2%, хоча при цьому ширина півстрічки зросла (з 4476 до 27850). Структуру оптимізованої матриці подано на рис. 6.2 праворуч.

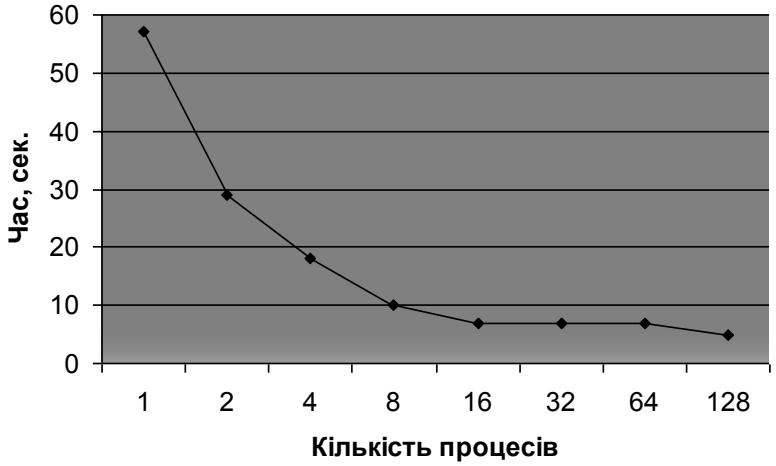


Рисунок 6.4 – Залежність часу розв’язування від кількості процесів для системи порядку 44436

Задача для всіх випадків завантажень розв’язувалася на кластері з залученням різної кількості процесів – від 1 до 256. Як виявилося, для розмірності даної задачі

максимально доступна кількість процесів вже не давала виграшу в часі. Найкращий показник (рис. 6.4) досягнуто у випадку розв’язування на 128 процесах, що в 11.73 разів менше, ніж час роботи послідовної програми.

6.2 Статичний розрахунок напруго-деформованого стану багатоповерхового житлового будинку.

Ефективне використання всіх доступних паралельних ресурсів досягається для задач більшої розмірності. До тих задач, для яких практично не досягнуто ефекту загинання кривої часу (тобто, такого стану, при якому збільшення числа процесів не зменшує часу розв’язування задачі), належить конструкція багатоповерхового житлового будинку (рис. 6.5). Конструкція розбита на 121761 скінченних елементів. Відповідна скінченно-елементна сітка містить 110265 вузлів.

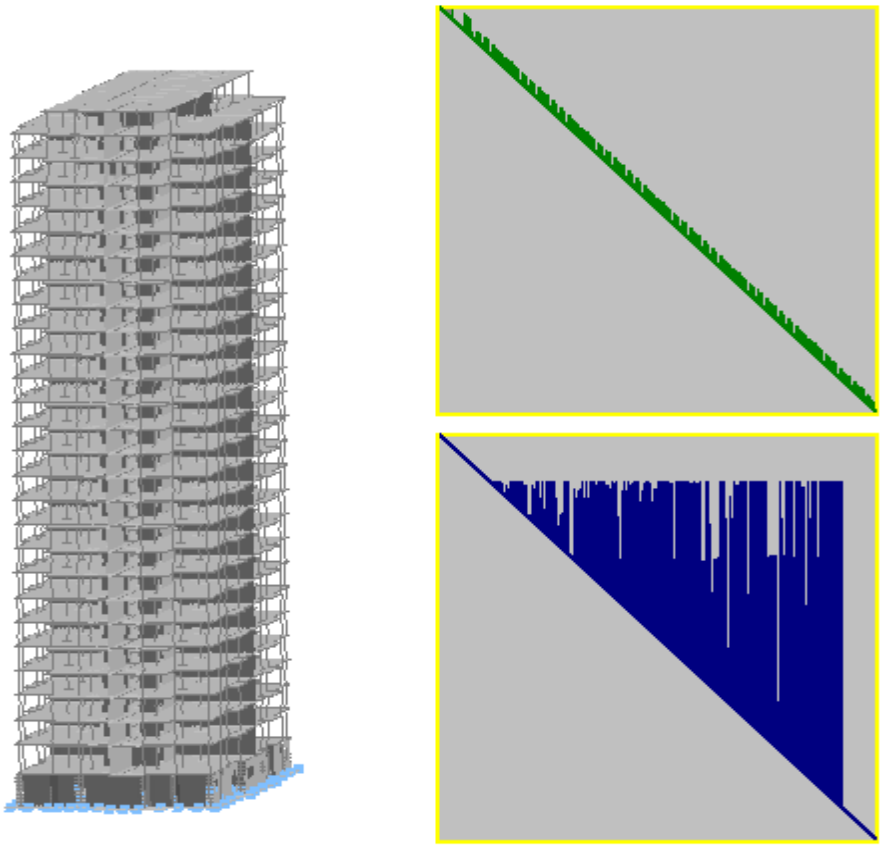


Рисунок 6.5 – Конструкція багатоповерхового житлового будинку та портрети відповідної матриці жорсткості до та після впорядкування елементів.

У результаті дискретизації отримано СЛАР порядку 661590. Вихідна структура матриці цієї системи мала щільність 5% і півширину стрічки 34242 (верхня частина

рис. 6.5). Після оптимізації структури матриці щільність впорядкованої структури складала 1% при півширині стрічки, рівній 541257 (нижня частина).

СЛАР розв’язано при різних кількостях процесів. Відношення часу розв’язування послідовним та паралельним алгоритмами на максимально доступній кількості процесів (256) склало 26.76 (рис. 6.6).

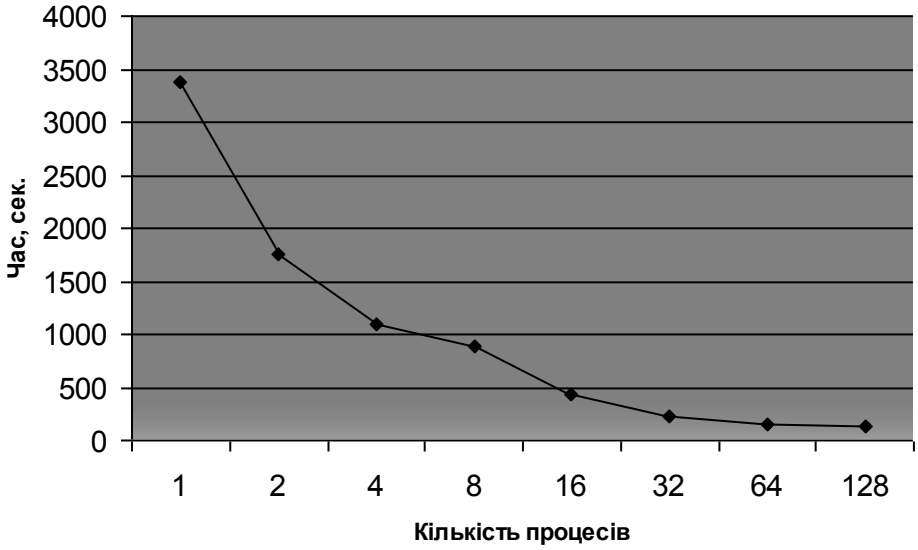


Рисунок 6.6 – Залежність часу розв’язування від кількості процесів для системи порядку 661590

Статичний розрахунок напруго-деформованого стану двовежового ділового центру.

Серед реальних задач, для розв’язування яких використовувалася розроблена програма, найбільша зі СЛАР з 5371727 рівняннями, стосувалася багатоповерхової споруди з двома вежами (рис. 6.7).

Конструкцію розбито на 972808 елементів, а скінченно-елементна сітка налічувала 895302 вузлів.

Об’єм даних виявився настільки великим, що єдиним можливим варіантом розв’язати задачу стало попереднє впорядкування ненульових елементів алгоритмом мінімальної степені, що зменшило вимоги до оперативної пам’яті до 76 Гб (у випадку матриці без оптимізації або використання інших способів оптимізації впорядкування, наприклад, фактор-дерев чи Катхіл-Макі, це число на порядок більше). Портрети матриці системи до та після оптимізації впорядкування

зображено у верхній та нижній частині рис. 6.8 відповідно. У результаті використання 256 вузлів вдалося скоротити час розв’язування задачі з 115 годин на одному процесорі до менш, ніж 5 годин (рис. 6.8).

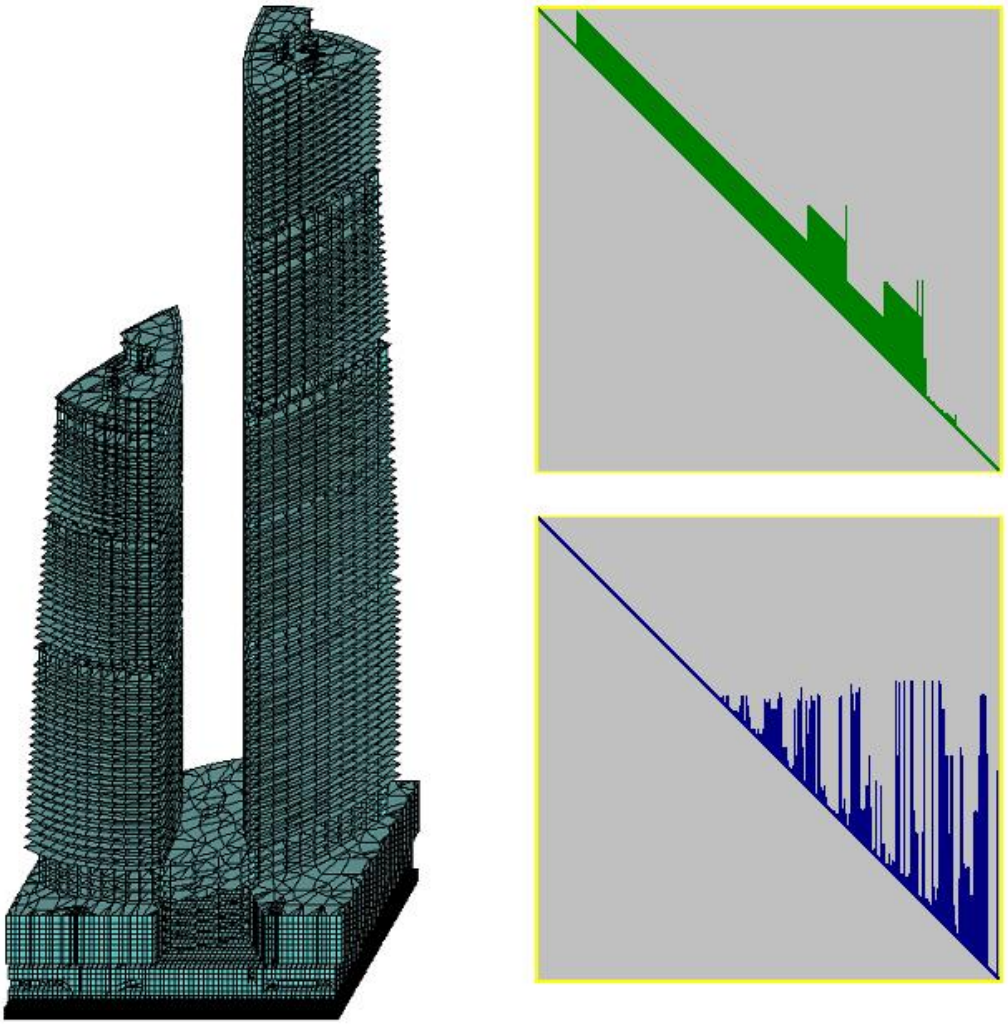


Рисунок 6.7 – Конструкція двовежевої споруди та портрети матриці жорсткості до та після оптимізації алгоритмом мінімальної степені.

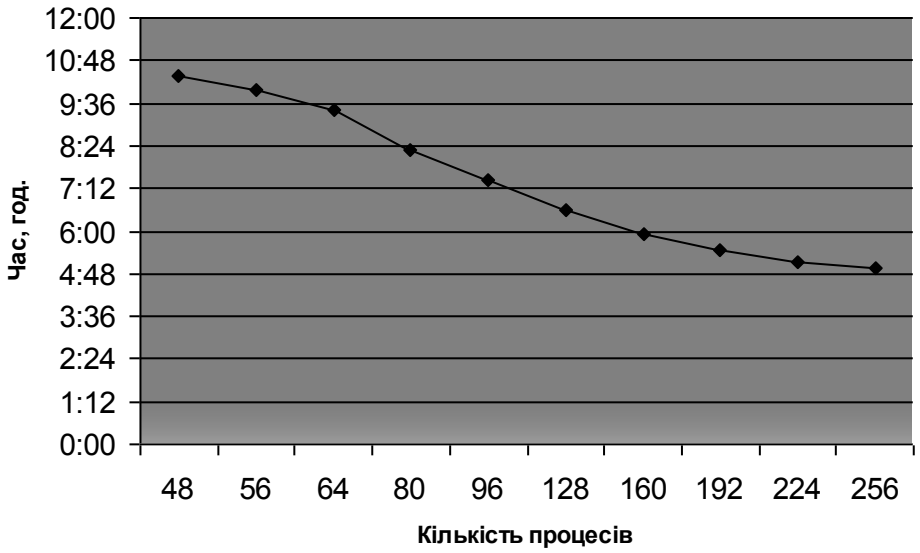


Рисунок 6.8 – Час розв’язування на різній кількості процесів для системи порядку 5371727

Для дослідження можливостей запропонованого паралельного алгоритму рішення СЛАР було вирішено ряд складних практичних завдань проектування - чисельні дослідження ряду об'єктів, в тому числі 27-ти поверхового будинку (рис. 6.8).

У табл. 6.1 наведені результати досліджень прискорень, одержуваних при вирішенні на комп'ютері гібридної архітектури СЛАР, що виникають при статичному аналізі міцності різних будівельних об'єктів і окремих конструкцій.

Таблиця 6.1 – Результати досліджень

Обєкт	Порядок матриці жорсткості	Півширина стрічки матриці	Час вирішення СЛАР	
			Ліра САПР** (хв.)	СКІТ_G* (хв.)
Будівля, 12 пов.	189 956	22 585	1	0,274
Фундамент	283 031	19 530	2	0,384
Балка, 3D СЕ	300 000	335	1	0,048
Будівля, 27 пов.	661 590	34 242	2	0,948
Плита, 3D СЕ	666 000	1 004	1	0,138
Плита, 3D СЕ	1 000 332	1 004	2	0,207
Балка, 3D СЕ	1 200 000	1 265	16	0,371

* СКІТ_G [14] – використано 1 GPU, гібридний алгоритм.

** ЛІра САПР [2] – багатопотова версія 2015 р., GPU не використовувався.

На рис. 4 представлена МСЕ 3D модель 27-ти поверхового будинку. Загальна характеристика і вхідні дані чисельної моделі просторових несучих конструкцій 27-ти поверхового будинку:

- сваї основи моделюються одновузловыми скінченими елементами (СЕ): Rz = 50990 *т/м*, Rx = 50990 *т/м*, Ry = 50990 *т/м*;
- фундаментна плита на відм. h = –4 *м* моделюється СЕ оболонки; модуль деформації бетону $E = 3{,}059\times10^6$ *т/м*^{2}; коефіцієнт Пуассона $\nu = 0{,}25$; товщина плити $H = 150$ *см*;
- колони моделюються стержньовими СЕ; $E = 3{,}059\times10^6$ *т/м*^{2}; $\nu = 0{,}25$; ширина и висота перерізу колон ($b\times h$) 25×90 *см*, 25×120 *см*.
- перекриття моделюються СЕ оболонки; $E = 3{,}059\times10^6$ *т/м*^{2}; $\nu = 0{,}25$; товщина плит перекриття $H = 18$ *см*.

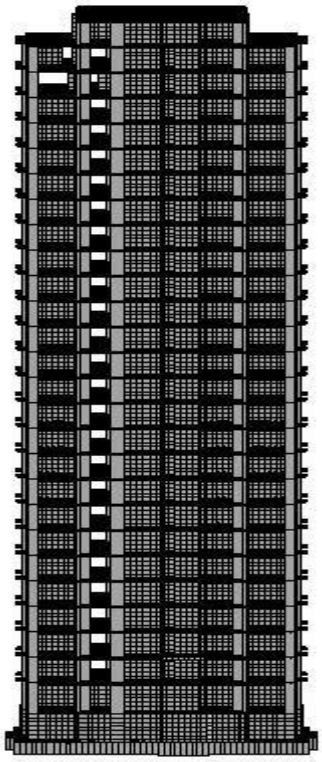


Рисунок 6.8 – Конструкція 27-ти поверхового будинку.

Розглянуто шість варіантів навантажень:

- власна вага 27-ти поверхового будинку;

- підлоги, перегородки, покрівля;
- корисні, сніг;
- трапециевидная навантаження від грунту на фундаментно-підвальну частину будівлі;
- вітер у напрямку X;
- вітер у напрямку Y.

Таким чином, роздільна СЛАР має 6 правих частин, тобто q = 6.

Деякі результати чисельного аналізу міцності висотної будівлі представлені детальними картинами розподілу переміщень по Y, Z (рис. 5) і зусиль N (рис. 6.10) для розглянутих варіантів навантажень.

Отримані результати чисельних досліджень поведінки будівельних конструкцій при статичних навантаженнях показали багаторазове скорочення часу рішення СЛАР з стрічковими симетричними позитивно певними матрицями на багатопроцесорних (багатоядерних) комп'ютерах з графічними прискорювачами при використанні запропонованих гібридних алгоритмів.

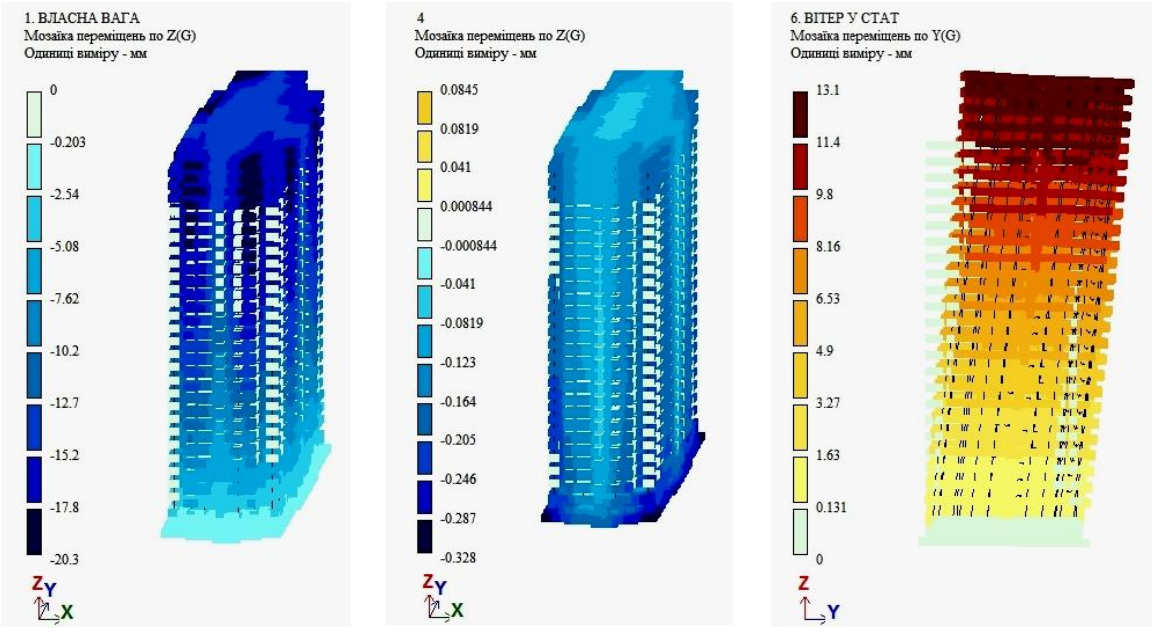


Рисунок 6.9 – Вертикальні переміщення будівлі по *Z*

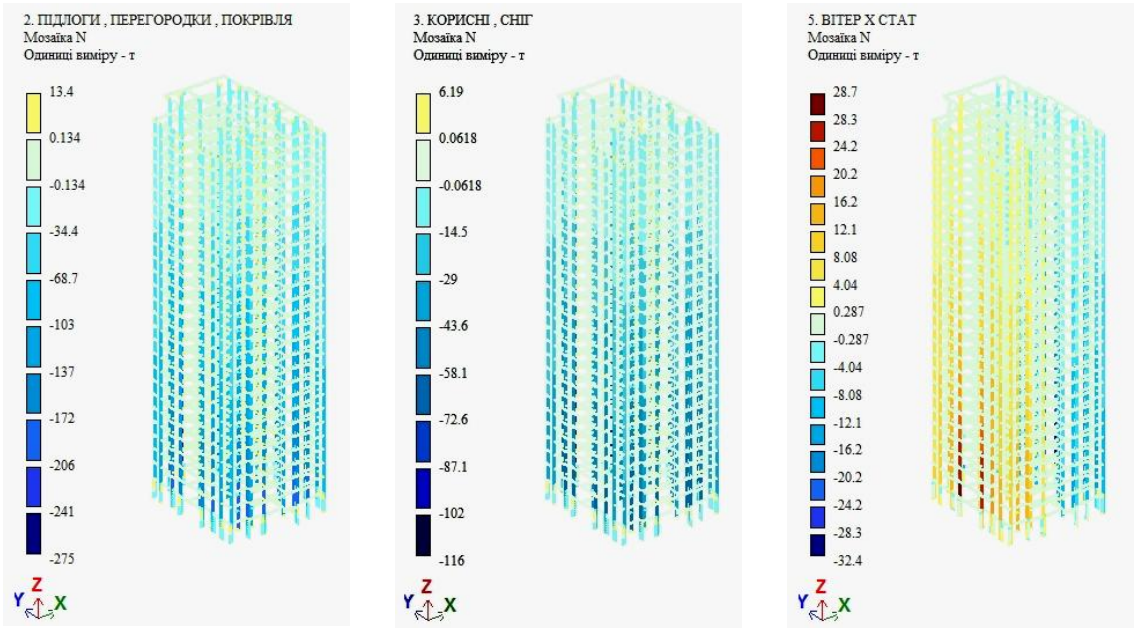


Рисунок 6.10 – Сили N, що діють на будівлю

Висновки до розділу

Загальна схема паралельних алгоритмів, а також її реалізація (розбиття на блоки, розподіл між процесорними пристроями, обробка тільки блоків, відповідних ненульовим блокам матриці розкладання) може бути успішно застосована і для вирішення СЛАР з іншими структурами розріджених матриць - профільної, блочно-діагональної з облямівкою і т.п. (Див., Наприклад, [15]).

Оптимізація обчислень для комп'ютерів гібридної архітектури лежить очевидно в такій модифікації алгоритмів з [8], яка дозволяє більш повно враховувати розріджену структуру матриці жорсткості і її LL^{T} -розкладання.

Проведена апробація запропонованих гібридних алгоритмів на вирішенні практичних завдань свідчить про перспективність цього напрямку в розвитку алгоритмічного і програмного забезпечення для математичного моделювання будівельних конструкцій.

Особливо істотний ефект прискорення можна очікувати при вирішенні практичних завдань проектування відповідальних особливо складних просторових конструкцій при статичних і динамічних навантаженнях, в т.ч. при дослідженні життєвого циклу висотних споруд з урахуванням нелінійних властивостей матеріалів конструкцій і грунтів підстав з використанням вищевикладених прямих методів вирішення СЛАР (з кількістю невідомих 10^6-10^9).

Використання наведених високопродуктивних паралельних алгоритмів дозволяє забезпечити високу якість проектування та безпеку зведення, експлуатації та реконструкції особливо складних будівельних об'єктів.

7 РОЗРОБКА СТАРТАП-ПРОЕКТУ

В поточному розділі описується можливість виходу стартап-проекту на ринок, також проводиться аналіз продукту та шляхи його впровадження.

7.1 Опис ідеї проекту

Наведемо опис ідеї стартап проекту, напрями його застосування, та переваги продукту для користувачів в таблиці 7.1.

Таблиця 7.1 - Опис ідеї стартап-проекту

Зміст ідеї стартап-проекту	Напрямки застосування	Вигоди для користувача
Аналіз міцності будівельної конструкції шляхом візуалізації визначення типу матриці жорсткості за допомогою нейронної мережі, а також розрахунок міцності будівельної конструкції найефективнішим алгоритмом.	1. Визначення міцності будівельної конструкції на етапі проекту	Визначення степеню міцності будівельної конструкції за інженерним проектом
	2.Аналіз міцності існуючої будівельної конструкції	Визначення степеню міцності будівельної конструкції за наявними дослідженнями та експериментами
	3.Визначення міцності будівельної конструкції при заданих наявних пошкодженнях	Визначення степеню міцності будівельної конструкції за наявними фізичними пошкодженнями
	4. Визначення максимально допустимих фактичних пошкоджень при експлуатації	Отримання рекомендацій, щодо недопущення перевищення експлуатації будівельної конструкції
	5. Система попередження потенційних елементів будівельної конструкції, що мають загрозу порушення міцності	Отримання рекомендацій потенційних елементів будівельної конструкції, що мають загрозу порушення міцності

Продовження таблиці 7.1

	6. Визначення максимально допустимих значень зовнішніх сил для непорушення міцності будівельної конструкції	Визначення максимально допустимих значень зовнішніх сил для непорушення міцності будівельної конструкції

Аналіз ринку показав, що по суті прямих конкурентів система немає, так як основна ідея даної системи покращення ефективності розрахунків шляхом визначення типу матриці жорсткості та вибору найефективнішого алгоритму для визначення міцності конструкції. Проте враховуючи те, що важливою складовою є також розрахунок міцності будівельних конструкцій, то є необхідність виділити таких конкурентів:

- EBuild — система автоматичного моделювання;
- ПК ЛІРА — система розрахунку міцності;
- RSTAB — система аналізу будівельних проектів.

В таблиці 7.2 наведено порівняння опису характеристик систем конкурентів та стартап-проекту.

Таблиця 7.2 - Сильні, слабкі та нейтральні характеристики ідеї проекту

№ п/п	Технікоекономічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	EBuild	ПК ЛІРА	RSTAB			
1	Вартість обслуговування	Відносно невелика	Не знайдено інформації	Не знайдено інформації	Не знайдено інформації		+	
2	Вартість експлуатації	Відносно невелика	Не знайдено інформації	Не знайдено інформації	Не знайдено інформації		+	

Продовження таблиці 7.2

3	Безвідмовність	За рахунок того, що система має багато рівнів а також, містить в собі композицію серверів ймовірність виходу всієї системи разом з ладу є дуже низькою	Дуже висока	Досить висока	Достатня		+	
4	Ремонтопридатність	За рахунок того, що система розгортається на декількох серверах, відсутня будь-яка проблема з ремонтом, адже ремонтувати прийдеться лише окремий сервер	Не знайдено інформації про архітектуру.	Не знайдено інформації	Не знайдено інформації		+	
5	Оплата праці	Беручи до уваги те, що дана система є стартапом, то на момент її запуску, вона буде розвиватись в більшій мірі за рахунок ентузіазму команди, і також можливих інвесторів, у вигляді майбутніх користувачів системи , а саме рекламодавців	Дуже висока	Висока	Висока			+

Продовження таблиці 7.2

6	Зручність користування	Розроблений дизайн на основі загальноприйнятих паттернів, та найкращих практик UX	Висока	Дуже висока	Нижче середнього			+
7	Простота освоєння	За рахунок того, що система розроблена на основі загальноприйнятих та звичних практик, простота освоєння є дуже високою	Висока	Середня	Висока		+	
8	Дизайн ресурсу	Не достатньо опрацьований, за рахунок не дуже високої відповідності сучасним тенденціям	Висока	Висока	Середня	+		
9	Відповідність патернами дизайну	Висока	Висока	Висока	Середня	+		
10	Відповідність тенденціям дизайну	Середня	Висока	Дуже висока	Середня	+		
11	Безпека даних користувача	Висока, так, як це одна з основних вимог до системи, що місять в собі дані про користувача	Дуже висока	Дуже висока	Середня		+	

Продовження таблиці 7.2

12	Відмово стійкість системи	Враховуючи те, що система має багато рівнів, та являє собою композицію серверів, і також має періодичні задачі з дамбуванням бази даних, то існує дуже низька ймовірність виходу всієї системи з ладу одчасно, з втратою якихось даних користувача тощо	Дуже висока	Дуже висока	Середня		+	
13	Підтримка оновлень	Присутня	Присутня	Присутня	Відсутня			

7.2 Технологічний аудит ідеї проекту

В таблиці 7.3 наведений технологічний аудит проекту, який показує список технологій та складових і також їх наявність, за допомогою яких реалізовується ідея проекту.

Таблиця 7.3 - Технологічна здійсненність ідеї проекту

№ п/п	Ідея стартап-проекту	Технології необхідні для реалізації	Наявність цих технологій	Доступність технологій
1	Збір даних для матриці жорсткості	Відкриті дані про стан будівельних конструкцій	Наявна	Доступна
2	Візуалізація матриці дорсткості	Використання алгоритмів візуалізації PLAN відносно заповненості таблиці	Наявна	Доступна

Продовження таблиці 7.3

3	Тренування нейроної мережі	Навчання нейронної мережі класифікації зображень на відомих візуалізованих матрицях різних типів	Потребує розробки	Використовує ті технології, та засоби, які є вільними для використання
4	Класифікація зображення матриці	Використання нейронної мережі для визначення типу матриці жорсткості за її візуалізацією	Потребує розробки	Використовує ті технології, та засоби, які є вільними для використання
5	Інформаційна система	Використання засобів розробки Front-end та Back-end частин	Потребує розробки	Використовує ті технології, та засоби, які є вільними для використання
Обрана технологія реалізації ідеї проекту: Надання користувачу рекомендацій алгоритму розвязання СЛАР, на основі визначення типу розрідженої матриці жорсткості нерегулярної структури				

Зважаючи на таблицю 7.3, технічна реалізація проекту є не зовсім простою, за рахунок того, що необхідно необхідно розробити деякі технології, проте на рахунок отримання готових бібліотек, та використання відкритих даних, жодних проблем виникнути не повинно.

7.3 Аналіз ринкових можливостей запуску стартап-проекту

Аналізючи ринкові можливості та загрози, можна більш ефективно спланувати, яким саме чином та в якому напрямі розвитку буде розвиватись проект, враховуючи стан поточного ринкового середовища, а також потреб клієнтів (таблиця 7.4).

Таблиця 7.4 - Попередня характеристика потенційного ринку

№ п/п	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	3

Продовження таблиці 7.4

2	Загальний обсяг продаж, грн/ум.од	
3	Динаміка ринку (якісна оцінка)	стагнує
4	Наявність обмежень для входу (вказати характер обмежень)	Присутні обмеження: 1. Необхідний рівень безпеки даних користувачів (GDPR) 2. Необхідний рівень відказостійкості системи, для збереження позицій 3. Необхідний рівень відповідності патернам UX, зручності використання та візуальної складової для збереження підтримки користувачів, а також зручності використання системи для людей з обмеженими можливостями
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні

Ринок рекомендацій є достатньо широкий, адже наявна велика кількість гравців, що займають дану нішу. Проте враховуючи те, що система в певній мірі є об'єднанням систем інших гравців, вона може отримати хорошу підтримку користувачів, за рахунок того, що в одній системі міститься вся необхідна інформація. В таблиці 7.5 наведено характеристику потенційних клієнтів стартап-проекту.

Таблиця 7.5 - Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Визначення міцності будівельних конструкцій	Інженери-будівельники та конструктори, що використовують	Різна поведінка може в основному залежати від різних типів	1.Збереження приватності даних, які користувачі надали системі.

Продовження таблиці 7.5

		всесвітню мережу Інтернет	будівельних конструкцій, їхньою діяльність, цільової задачі конструкції	2.Надання правильних, релеватних висновків
2.	Оцінка пошкоджень та аварійності будівель	Працівники установ, що контролють та перевіряють стан будівельних конструкцій	Різна поведінка, яка залежить від сфери діяльності	1. Надання загально-статистичної інформації про різні класи будівель, для можливості надання релевантної оцінки, що дасть змогу визначення стану будівельної конструкції. 2.Надання інструменту створення оцінок аварійності об'єктів 3.Надання інструменту перевірки якості будівельних конструкцій

Після формування списку потенційних можливих клієнтів, проведено аналіз ринку, та наведено список факторів, що можуть перешкодити успішному впровадженню системи. Дані фактори наведено в таблиці 7.6.

Таблиця 7.6 - Фактори загроз стартап-проекту

№ п/п	Фактор загрози	Зміст загрози	Можлива (необхідна) реакція компанії
1	Конкуренти, уже давно знаходяться на ринку, та встигли проявити свої сильні сторони	Складно позиціювати себе, як продукт, що є кращим за їхній. Також існує проблема з отриманням ТОПових місць в результатах пошуку пошукових систем	Сильна маркетингова компанія, з залученням підрядчиків, SEO оптимізація
2	Низька довіра та зацікавленість користувачів до нової	На момент запуску системи, до того часу, поки вона не проявить	Створення продукту з мінімальною кількістю багів, який буде

Продовження таблиці 7.6

	системи	себе, як принаймні рівну конкурентам, буде присутня недовіра до нового продукту	максимально якісним, та матиме переваги в порівнянні з конкурентами. Сильна маркетингова компанія, що покаже переваги продукту
3	Введення змін, що стосуються оброки та зберігання персональних даних користувачів	Можливе зниження якості рекомендацій, за рахунок втрати частини інформації про користувачів	Відсутня

Також складено перелік факторів, що позитивно впливають на можливість успішного впровадження системи (таблиця 7.7).

Таблиця 7.7 - Фактори можливостей стартап-проекту

№ п/п	Фактор можливостей	Зміст можливості	Можлива реакція компанії
1	Отримання відгуку від користувачів, щодо системи	Покращення продукту за рахунок врахування відгуку реальних користувачів, щодо їх досвіду з роботи з системою	Аналіз, врахування та реалізація нової функціональності системи, якщо це потрібно
2	Покращення роботи нейронної мережі	Покращення визначення типу матриці жорсткості ляхом дотренувовування нейронної мережі	Створення аналітики та метрик, для аналізу поведінки користувачів, та власне сам аналіз
3	Розширення системи	Розширення системи, за рахунок впровадження нового функціоналу, що враховує отриманні дані про користувача	Побудова плану розвитку системи

Проведено аналіз поточного ринку, та сформовано риси конкуренції на ринку (таблиця 7.8).

Таблиця 7.8 - Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Тип конкуренції - монополія	Монополія, так як сформовані та існують чіткі лідери в різних напрямах інтересів	Розширяти сферу впливу за рахунок врахування українського ринку
2. За рівнем конкурентної боротьби - інтернаціональна	Інтернаціональний, так як виділені монополісти є інтернаціональними компаніями	Своєчасний вихід на інтернаціональний ринок, лише після того, коли буде сильна підтримка на локальному
3. За галузевою ознакою - внутрішньогалузева	Внутрішньогалузева, так як всі системи, що надають рекомендації є інтернет ресурсами	Врахування розвитку сфери, та слідування її сучасним тенденціям
4. За характером конкурентних переваг - нецінова	Нецінова, так, як всі системи не передбачають здійснення жодних платежів користувачем	Конкурування за рахунок надання більш релевантних висновків та аналізів, а також врахування відгуків користувачів системи, для її покращення

Також було проаналізовано конкурентність в середині галузі за М.Портером та наведено в таблиці 7.9.

Таблиця 7.9 - Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	EBuild ПК ЛІРА	Проаналізувавши ринок, можна побачити, що не	Немає необхідності	Клієнтам може не подобатись робота деяких модулів	Майже відсутні

Продовження таблиці 7.9

	RSTAB	існує бар’єру для виходу на ринок		системи	
Висновки:	Пряма конкуренція майже відсутня	Можлива поява прямих конкурентів	Постачальники на ринок не впливають	Клієнти дають відгук щодо роботи системи	Майже відсутні

Як видно конкуренція існує, проте тим не менш ринок є привабливим для того, щоб впроваджувати систему. Проаналізувавши конкуренцію, що була наведена в таблиці 7.9, та вимоги користувачів до системи (таблиця 7.5), було сформовано набір ключових факторів конкурентоспроможності системи (таблиця 7.10) та обґрунтовано чому вони є значущими.

Таблиця 7.10 - Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Визначення загальної міцності будівельної конструкції	Системи націлені на надання рекомендацій всередині конкретної сфери (наприклад лише для залізно-бетонних виробів). Дана ж система надає можливість визначення міцності будь якої будівельної конструкції. А також визначає міцність будівельної конструкції найефективнішим алгоритмом
2	Можливість мінімізування апаратних ресурсів шляхом оптимізації роботи за рахунок аналізу матриці жорсткості та вибору ефективних алгоритмів	
3	Рекомендації на основі моделі користувача та стану будівельної конструкції	Рекомендаційні системи надають рекомендації на основі тих даних, що були введені безпосередньо в їхній системі, наприклад на основі раніше переглянутих обєктів тощо. Дана ж система враховує стан будівельної конструкції та тип матриці жорсткості, для можливості надання користувачу рекомендацій відповідно до його класів.
4	Безкоштовність системи для користувача	Дана система є безкоштовною для користувача, проте йому надається також і реклама, але яка при цьому є релевантною і відповідає до його класу та інтересів.

Продовження таблиці 7.10

5	Можливість отримання опису соціальних груп, для надання рекламних пропозицій	Рекламодавець має змогу отримати статистичну інформацію певної групи людей, та сформувати свою рекламу таким чином, щоб її конверсія була максимальною. Дана функція є платною.
6	Закладена можливість розширення системи	Архітектура система побудована таким чином, що легко піддається розширенню, та масштабуванню. При цьому оброблені дані можна також використовувати для розроблення інших застосунків, що базуються на інтересах користувачів.

За сформованими факторами конкурентоспроможності, було проведено аналіз конкурентів, та виявлено сильні та слабкі сторони системи стартап-проекту (таблиця 7.11).

Таблиця 7.11 - Порівняльний аналіз сильних та слабких сторін

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з системою рекомендацій на основі поведінкової моделі користувача						
			–3	–2	–1	0	+1	+2	+3
1	Визначення загальної міцності будівельної конструкції	20				+			
2	Можливість мінімізування апаратних ресурсів шляхом оптимізації роботи за рахунок аналізу матриці жорсткості та вибору ефективних алгоритмів	20		+					
3	Рекомендації на основі моделі користувача та стану будівельної конструкції	20			+				
4	Безкоштовність системи для користувача	15					+		

Продовження таблиці 7.11

5	Можливість отримання опису соціальних груп, для надання рекламних пропозицій	15					+		
6	Закладена можливість розширення системи	10					+		

Також за сформованими слабкими та сильними сторонам поточного стартап-проекту, проаналізувавши ринок та визначивши загрози а також можливості, які є на даний момент, та можуть виникнути в майбутньому, було складено SWOT-аналіз (таблиця 7.12), для того, щоб оцінити внутрішні та зовнішні фактори, які можуть якимось чином вплинути на розвиток проекту чи на його впровадження.

Таблиця 7.12. SWOT-аналіз стартап-проекту

Внутрішні фактори	
Сильні сторони	Слабкі сторони
Автоматизована система Можливість отримання опису будівельної конструкції Можливість оптимізації апаратних ресурсів Закладена можливість розширення системи Безкоштовність системи для користувача	Слабкий дизайн Висока вартість обслуговування в порівнянні з конкурентами
Зовнішні фактори	
Можливості	Загрози
Отримання фідбеку від користувачів, щодо системи Покращення роботи нейронної мережі Розширення системи	Конкуренти, уже давно знаходяться на ринку, та встигли проявити свої сильні сторони Низька довіра та зацікавленість користувачів до нової системи Введення змін, що стосуються оброки та зберігання персональних даних користувачів

На основі SWOT-аналізу, було розроблено комплекс заходів (альтернатив ринкового впровадження), які орієнтовані на те, щоб дати змогу стартап-проекту вийти успішно на ринок за оптимальний час. Комплекс заходів наведено в таблиці 7.13.

Таблиця 7.13. Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Виведення робочої системи з робочою основною функціональністю системи проте з мінімальним дизайном	Ресурси будуть отриманні за рахунок інвесторів	3 місяці
2	Покращення дизайну та системи в цілому на основі відгуку користувачів щодо системи	Відсутня	1 місяць
3	Виведення робочої системи з готовим інструментом для рекламодавців для створення рекламних пропозицій, для отримання коштів, що будуть використання для наступного розвитку системи	Ресурси будуть отримані за рахунок оплати інструментів рекламодавцями	2 місяці

7.4 Розроблення ринкової стратегії проекту

Для ефективного виходу на ринок є необхідність сформувати список цільових груп споживачів (таблиця 7.14) а також обрати, ті цільові групи, які надають більше можливостей для стартап-проекту.

Таблиця 7.14 - Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Малий бізнес	Низька готовність	Низький попит	Збільшення конкуренції з часом	Висока складність

Продовження таблиці 7.14

2	Середній бізнес	Середня готовність	Середній попит	Збільшення конкуренції з часом	Середня складність
3	Великий бізнес	Висока готовність	Середній-вище середнього	Збільшення конкуренції з часом	Середня складність
4	Звичайні користувачі мережі Інтернет	Висока готовність	Високий попит	Збільшення конкуренції з часом	Середня та нижче середнього
Які цільові групи обрано: Для найкращої реалізації стартап-проекту враховуючи простоту входу в сегмент та орієнтовний попит найдоцільніше обрати 2 та 3 групу					

Для здійснення максимально ефективної роботи в обраних сегментах було сформовану базову стратегію розвитку (таблиця 7.15), і також базову стратегію конкуретної поведінки (таблиця 7.16).

Таблиця 7.15 - Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Виведення робочої системи з основною функціональністю системи проте з мінімальним дизайном	Стратегія диференціації	Релевантніші рекомендації ніж в інших системах; швидкість розширення системи	Надання користувачу можливостей, що відрізняються від конкурентів, та робить дану систему актуальнішою

Таблиця 7.16 - Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохід цем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Частково	Компаній буде шукати як нових споживачів, так і забирати існуючих у конкурентів	Так, якщо будуть покращені алгоритми класифікації	Стратегія лідера

Враховуючи вимоги споживачів до продукту, а також обрані стратегії розвитку та конкурентної поведінки розроблена стратегія позиціонування (таблиця 7.17), яка буде притаманна продукту на ринку, для можливості його максимальної реалізації та його однозначної ідентифікації.

Таблиця 7.17 - Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспромо жні позиції власного стартаппроекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Збереження приватності даних користувачів	Стратегія лідерства по витратах	Оптимізація	Високий рівень збереження даних користувача Отримання релевантних рекомендацій Високий рівень кастомізації системи
2	Надання релевантних висновків та рекомендацій	Стратегія спеціалізації	Точність рекомендацій	
3	Надання інструменту створення, та управління об'єктами	Стратегія спеціалізації	Високий рівень кастомізації	

7.5 Розроблення маркетингової програми стартап-проекту

Було проаналізовані вимоги потенційних користувачів стартап-проекту, та обрано ті ключові вимоги (таблиця 7.18), які є необідними при запуску стартап-проекту, для успішного виходу на ринок.

Таблиця 7.18 - Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Збереження приватності даних користувачів	Збереження даних в хешованому вигляді (в перспективі в Azure Storage)	Хмарне збереження даних, надійне, безпечне та витримує високі навантаження
2	Надання релевантних рекомендацій	Додаткові модулі у вигляді модулів застосунку	Високий показник релевантності рекомендацій
3	Надання інструменту створення, та управління об'єктами	Додаткові модулі у вигляді модулів застосунку	Високий рівень кастомізації інструментів

Також описано три рівні моделі товару (таблиця 7.19).

Таблиця 7.19 - Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Застосунок, який візуалізує матрицю жорсткості будівельної конструкції, за даною візуалізацією визначає тип матриці жорсткості та вибір ефективного алгоритму для визначення міцності будівельної конструкції</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Високий показник релевантності рекомендацій
2. Зручний дизайн
3. Швидка роботи клієнтської частини</td><td>М</td><td>Вр/Тх/Тл</td></tr>
<tr><td colspan="3">Програма пройшла тестування та відповідає необхідним параметрам якості для виходу на ринок. Також наявна документація</td></tr>
<tr><td colspan="3">Застосунок, що дає змогу працювати з системою</td></tr>
<tr><td colspan="3">Марка: TESTConsole</td></tr>
<tr><td>III. Товар із підкріпленням</td><td colspan="3">Програмне забезпечення</td></tr>
<tr><td colspan="4">Товар буде захищено від копіювання за рахунок закритого коду</td></tr>
</table>

Важливим етапом є визначення цінових меж стартап-проекту (таблиця 7.20). Так, як рекомендаційна система отримує прибуток за рахунок реклами, то й ціна буде рахуватись за одиницю реклами, на яку користувач перейшов.

Таблиця 7.20 - Визначення меж встановлення ціни

№ п/п	Рівень цін на товаризамінники	Рівень цін на товариаподатки	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	10000 грн / перевірка	15000 грн / перевірка	Високий рівень доходів	Так, як система має надати користувачам лише релевантну рекламу, то її конверсія буде значно вищою ніж в конкурентів, то мінімальна ціна — 1.01 / клік, максимальна ціна — 25000 грн / пеервірка

Також проаналізовано ринок та системи конкурентів, та сформовано систему збуту (таблиця 7.21) та концепцію маркетингових комунікацій (таблиця 7.22)

Таблиця 7.21 - Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Придбання доступу до каталогу інтересів та класів	Постачальник відсутній	Виробник-споживач	Застосунок

Таблиця 7.22 - Концепція маркетингових комунікацій

Специфіка поведінки обраних цільових клієнтів	Канали комунікацій, що використовують цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Знають, що їм потрібно тут і зараз	Веб-сайти, соціальні мережі, телефон, зустрічі	Релевантні рекомендації та таргетовані пропозиції	Надання користувачу списку переваг системи над конкурентами	Надання статистичної інформації про користувачів

Висновки до розділу

Даний програмний застосунок націлений для визначення та аналізу міцності будівельної конструкції. Функціональність авторизованих користувачів, а саме інженерів конструкторських бюро дозволяє отримувати статистичні дані та створювати нові проекти будівельних конструкцій та обраховувати їх міцність основна.

Так як системи визначення міцності будівельних конструкцій є популярними в наш час, існує конкуренція, проте вона не є прямою. Зважаючи на це, ринок є привабливим для освоєння. Проте існує імовірність появи нових конкурентів, які будуть намагатись створити подібну систему. Для того, щоб захистити проект від копіювання, весь код системи буде закритим. Технології, що були обрані при розробці знаходяться в відкритому доступі.

Основною перевагою системи є функціональність надання рекомендацій нових інтересів на основі поведінки користувача в соціальних мережах, а також якість та зручність роботи в самій системі.

Також зважаючи на групи потенційних клієнтів, а саме великий бізнес є дуже хороші перспективи для виходу на ринок, за рахунок високого рівня доходів цільової групи споживачів. При цьому було обрано стратегію диференціації, як базову стратегію розвитку, що дасть зможу в майбутньому все більше залучати й інші групи споживачів.

Отже, зважаючи на переваги та недоліки системи, стартап-проект має успішно вийти на ринок, та завоювати велику кількість споживачів.

ВИСНОВКИ

У даній магістерській дисертації було створено програму для вирішення слар з розрідженими матрицями на основі нейромережі. Програмна реалізація системи написана мовою програмування Python на платформі PyQt на базі операційної системи Windows. Такий вибір зроблено на користь використання розвиненого середовища розробки Qt та легкої можливості портування програми на будь яку іншу платформу, зокрема варто відмітити наявність зручних статистичних функцій та функцій створення різноманітних графіків і інших методів, що дозволяють задовольнити вимогу створення зручного графічного інтерфейсу користувача.

Першим етапом розробки цього проекту був аналіз предметної області. Були досліджені останні теоретичні роботи у даній сфері, виокремлено та використано в роботі результати найбільш перспективних із них. На основі проведених досліджень стало можливим зробити обґрунтований вибір найбільш ефективних для даної задачі алгоритмів для різних типів розріджених матриць нерегулярної структури.

Була розроблена нейромережа для визначення типу розрідженої матриці нерегулярної структури та вибору оптимального алгоритму розвязку СЛАР.

Було проведено дослідження дослідження прямих паралельних алгоритмів для СЛАР з розрідженими матрицями нерегулярної структури з наближеними даними

Була проведена аналіз та апробація алгоритмів для математичного моделювання в прикладних задачах.

В рамках опису архітектури побудованої системи розглянуто перелік усіх її модулів, наведено діаграми компонентів, що можуть бути утворені в результаті динамічного зв'язування компонент. Таким чином в результаті проведеної роботи була досягнута поставлена мета дослідження.

За матеріалами дисертації було опубліковано 3 наукові роботи: 1 стаття [35] та 2 тез доповідей на конференціях [36; 37]

ПЕРЕЛІК ПОСИЛАНЬ

1. E.A. Velikoivanenko, A.S. Milenin, A.V. Popov, V.A. Sidoruk, A.N. Khimich. Methods and technologies of parallel computing for mathematical modeling of stress-strain state of constructions taking into account ductile fracture. // Journal of Automation and Information Sciences, Vol. 46, Issue 11, 2014, P. 23–35.

2. Блатов И.А. Методы решения систем с разреженными матрицами / Блатов И.А., Глушакова Т.Н., М.Е. Эскаревская – Воронеж: Воронежский гос. ун-т, 2002. – 34 с.

3. Брамеллер А. Слабозаполненные матрицы / Брамеллер А., Аллан Р., Хэмэм Я. – М.: Энергия, 1979. – 192 с.86

4. Глушакова Т.Н. Методы работы с разреженными матрицами произвольного типа / Глушакова Т.Н. Эксаревская М.Е. – Воронеж: Воронежский гос. ун-т, 2005. – 44 с.

5. Глушакова Т.Н. Методы решения систем с разреженными матрицами / Глушакова Т.Н., Блатова И.А. – Воронеж: Воронежский гос. ун-т, 2000. – 36 с.

6. Годунов С.К. Решение систем линейных уравнений / Годунов С.К. – Новосибирск: Наука, 1980.

7. Недашковський М.О. Обчислення з λ-матрицями / Недашковський М.О., Ковальчук О.Я. – К.: Наук. думка, 2007. – 294 с.

8. Джордж А. Численное решение больших разреженных систем уравнений / Джордж А., Дж. Лиу. – М.: Мир, 1984. – 390 с.

9. Джордж А., Лиу Дж. Численное решение больших разреженных систем уравнений. – М.: Мир, 1984. – 333 с.

10. Дж. Голуб, Ч. Ван Лоун Матричные вычисления: Пер. с англ. – М.:Мир, 1999.

11. Р. Тюарсон, Разреженные матрицы: Пер. с англ. – М.: Мир, 1977.-192.с

12. Chow E. ILUS: an incomplete LU Preconditioner in Sparse Skyline format / Chow E., Saad Y. – In: Int. J. for Num. methods.

13. Компьютерное моделирование: моделирование как метод научного познания [Електронний ресурс] // Режим доступу: http://www.econf.rae.ru/article/6722

14. Saad Y. ILUT: a dual threshold incomplete ILU factorization / Saad Y. // Report umsi-92-38, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, 1992, - 18 p

15. Ильин В.П. Экспериментальный анализ явных методов неполной факторизации / Ильин В.П., Ицкович Е.А. // Сборник научных трудов «Численные методы и математическое моделирование» под ред. Ильина В.П. – Новосибирск: ВЦ СО АН СССР – 1990, с. 85-94.

16. Эстербю О. Прямые методы для разреженных матриц / О. Эстербю, З. Златев – М.: Мир, 1987. – 118 с.

17. SUPERLU, a sequential library for the direct solution of large, sparse, nonsymmetric systems of linear equations on high performance machines [Електронний ресурс] / Sherry Li, Jim Demmel, John Gilbert, Laura Grigori. – 2008. – Режим доступу : http://crd.lbl.gov

18. Naumov M. Parallel Incomplete-LU and Cholesky Factorization in the Preconditioned Iterative Methods on the GPU. – 19 p. – [Електронний ресурс] Режим доступу: https://research.nvidia.com/sites/ default/files/publications/nvr-2012-003.pdf

19. Эстербю О. Прямые методы для разреженных матриц / О. Эстербю, З. Златев – М.: Мир, 1987. – 118 с.

20. Ильин В.П. Экспериментальный анализ явных методов неполной факторизации / Ильин В.П., Ицкович Е.А. // Сборник научных трудов «Численные методы и математическое моделирование» под ред. Ильина В.П. – Новосибирск: ВЦ СО АН СССР – 1990, с. 85-94.

21. SUPERLU, a sequential library for the direct solution of large, sparse, nonsymmetric systems of linear equations on high performance machines [Електронний ресурс] / Sherry Li, Jim Demmel, John Gilbert, Laura Grigori. – 2008. – Режим доступу : http://crd.lbl.gov

22. Naumov M. Parallel Incomplete-LU and Cholesky Factorization in the Preconditioned Iterative Methods on the GPU. – 19 p. – [Електронний ресурс] Режим доступу: https://research.nvidia.com/sites/ default/files/publications/nvr-2012-003.pdf

23. Перельмутер А.В., Фиалко С.Ю.Прямые и итерационные методы решения большеразмерных конечно-элементных задач строительной механики // Киев, Киевский национальный технический университет строительства и архитектуры - 2009, 6с.

24. Сушко Г.Б. Многопоточная параллельная реализация итерационного алгоритма решения систем линейных уравнений с динамическим распределением нагрузки по нитям вычислений / Г.Б. Сушко, С.А. Харченко – 2009, 6с.

25. Городецкий А.С. Компьютерные модели конструкции / Городецкий А.С., Евзеров И.Д. – К.: Факт, 2005. – 334 с.

26. Михалевич В.С. Численные методы для многопроцессорного вычислительного комплекса ЕС / Михалевич В.С., Бик Н.А., Брусникин Б.Н.,…, Химич А.Н. и др./ Под редакцией И.Н. Молчанова.– М.: Издание ВВИА им. проф. Н.Е. Жуковского, 1986. – 401 с.

27. Молчанов И.Н. Некоторые методы решения больших разреженных систем уравнений на многопроцессорном вычислительном комплексе (МВК) / И.Н. Молчанов, И.В. Решетуха, О.В. Рудич, С.П. Семенченко. – К., 1991. – 31 с. – (Препр. / АН УССР. Ин-т кибернетики им. В.М. Глушкова; 91-2).

28. Intel® Math Kernel Library (Intel® MKL) – Режим доступу: https://software.intel.com/en-us/intel-mkl

29. Garbow B.S. Matrix Eigensystem Routine / Garbow B.S., Royle J.M., Dongarra J.J., Moller M.M. – EISPACK Guide Extension Lecture Notes in Computer Science, vol. 51. Springer-Verlag, 193. – 236 p

30. Химич А.Н. Численное программное обеспечение MIMD– компьютера Инпарком / Химич А.Н., Молчанов И.Н.,… Полянко В.В., и др. – Киев: Наукова думка, – 2007. – 222 с.

31. Chow E. ILUS: an incomplete LU Preconditioner in Sparse Skyline format / Chow E., Saad Y. – In: Int. J. for Num. methods.

32. Saad Y. ILUT: a dual threshold incomplete ILU factorization / Saad Y. // Report umsi-92-38, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, 1992, - 18 p.

33. Alfredo Buttari, Julien Langou, Jakub Kurzak, and Jack Dongarra: A Class of Parallel Tiled Linear Algebra Algorithms for Multicore Architectures.Parallel Computing, Volume 35, Issue 1, P. 38-53, 2009, ISSN:0167-8191.

34. Капорин И.Е. Постфильтрация множителей IC2-разложения для балансировки параллельного предобусловливания / Капорин И.Е., Коньшин И.Н. // Журнал вычислительной математики и математической физики. – 2009. – Т. 49, № 6. – С.940-957.

35. Марочканич О.Р. Інтелектуалізація обчислень для задач математичного моделювання складних процесів і об’єктів / В.А. Сидорук, П.С. Єршов, Д.О. Богурський, О.Р. Марочканич // Журнал «Комп’ютерна математика». – 2019. – № 1. – С.143-150., ISSN:2616-938X

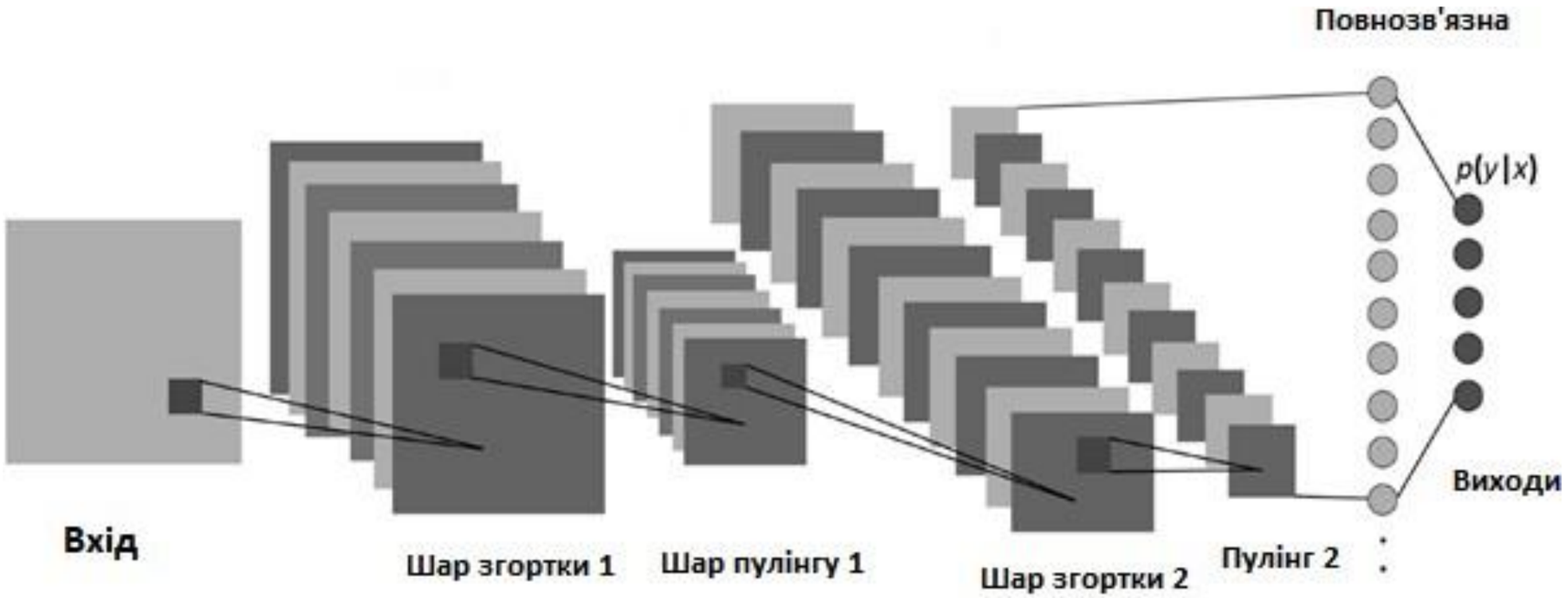
36. Марочканич О.Р. Технології обчислень для визначення міцності будівельних конструкцій / О.Р. Марочканич // Матеріали Міжнародної наукової інтернет-конференції «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення». – м.Тернопіль, 11 червня 2019 р. – С.48-50.

37. Марочканич О.Р. Інтелектуалізація обчислень для задач розрахунку міцності конструкцій / О.Р. Марочканич, О.М. Хіміч // Матеріали ІІІ всеукраїнської науково-практичної конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2019) – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 20-22 листопада 2019 р. – С.. – С.67-71.

ДОДАТОК А

ГРАФІЧНИЙ МАТЕРІАЛ

СТРУКТУРА ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ



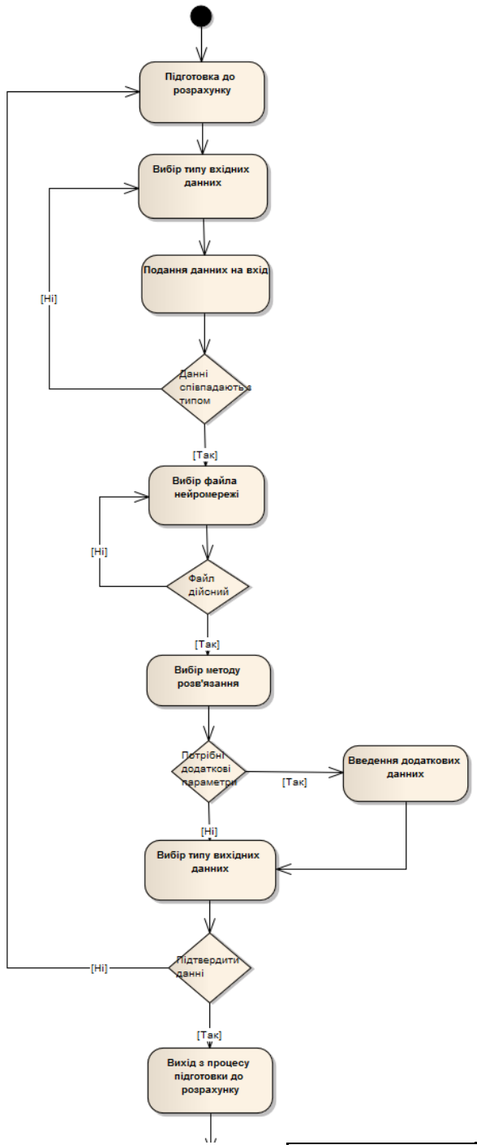
Демонстраційний плакат до магістерської дисертації

на тему «Інтелектуалізація обчислень для задач розрахунку міцності конструкцій»

Виконав студент гр. ІС-82мп Марочканич Олександр Романович

Керівник Хіміч Олександр Миколайович

СХЕМА СТРУКТУРНА ДІЯЛЬНОСТІ



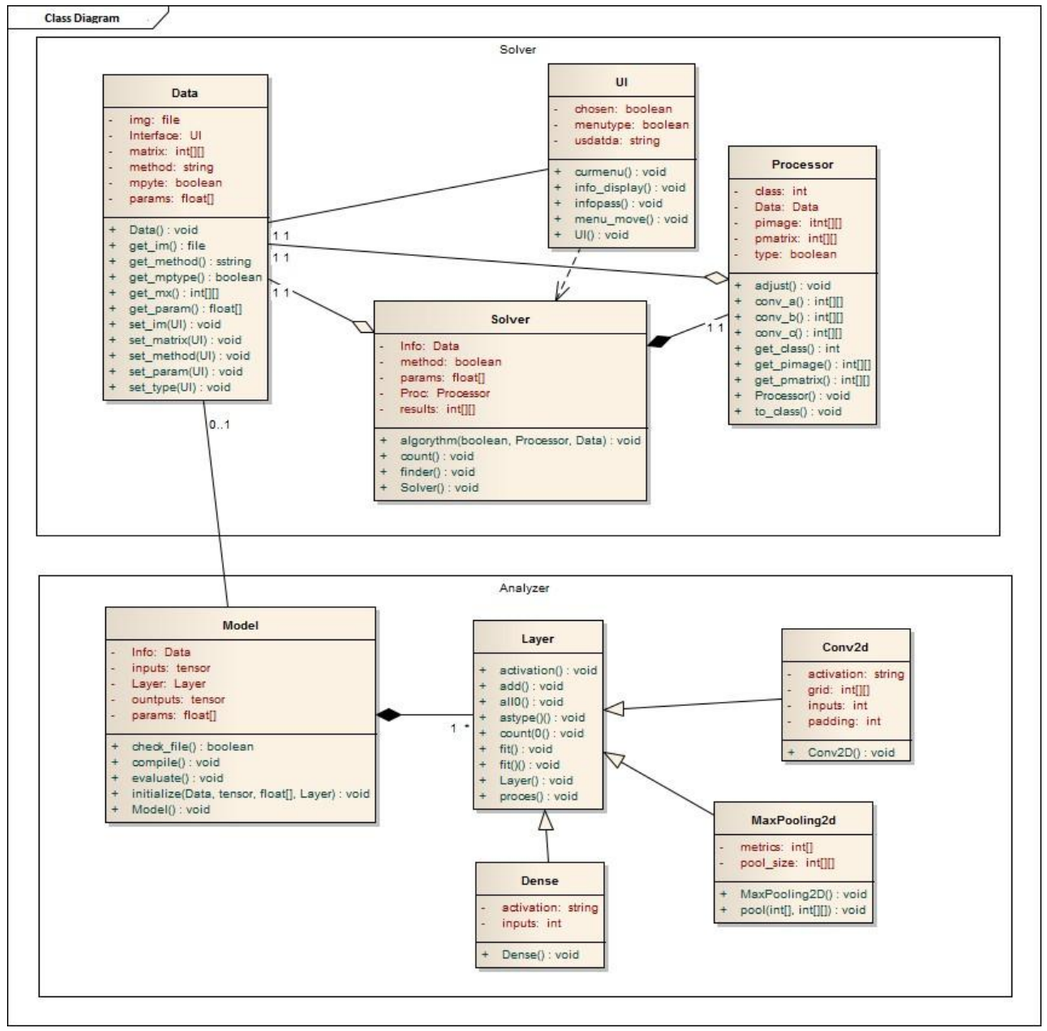
Демонстраційний плакат до магістерської дисертації

на тему «Інтелектуалізація обчислень для задач розрахунку міцності конструкцій»

Виконав студент гр. ІС-82мп Марочканич Олександр Романович

Керівник Хіміч Олександр Миколайович

СХЕМА СТРУКТУРНА КЛАСІВ



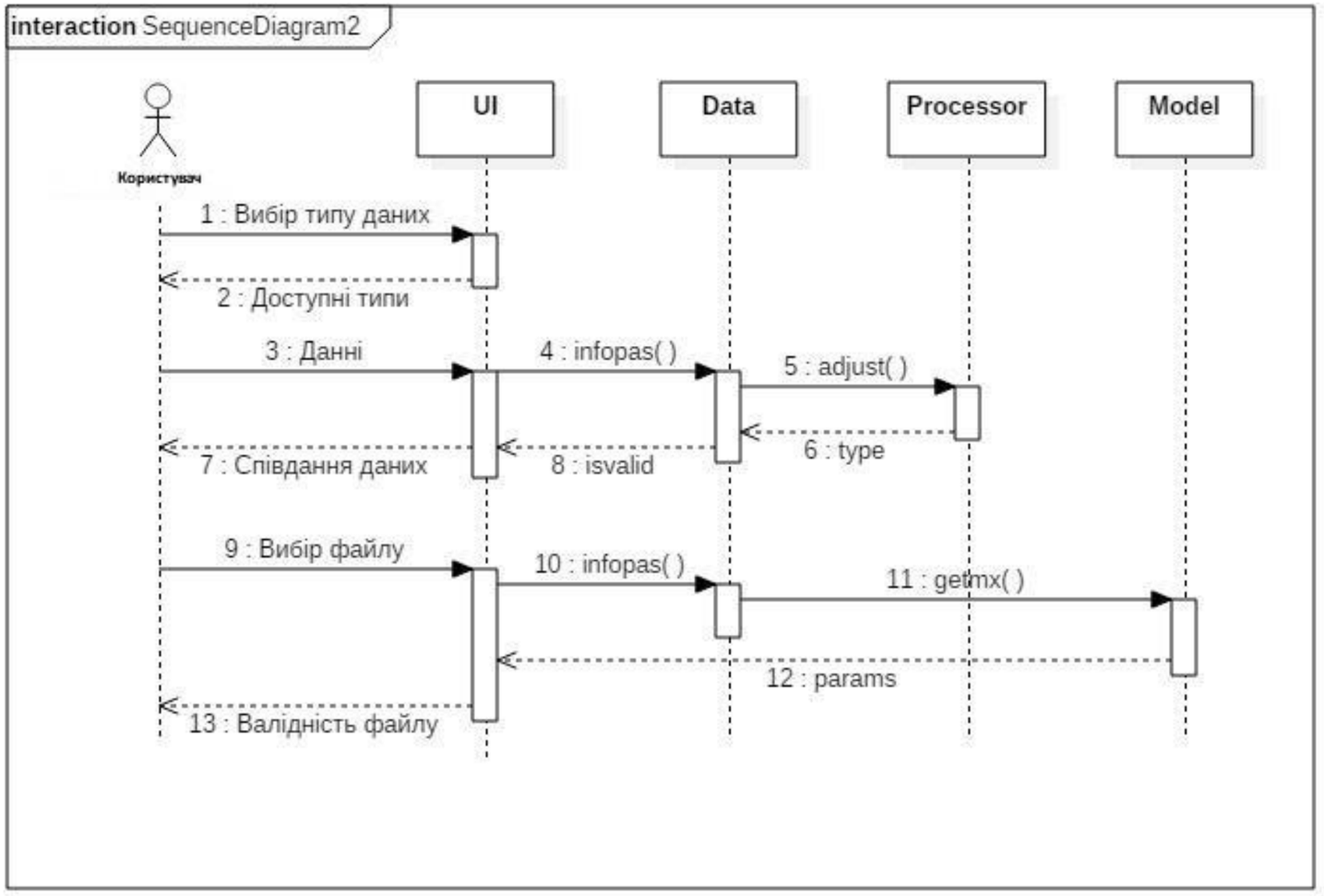
Демонстраційний плакат до магістерської дисертації

на тему «Інтелектуалізація обчислень для задач розрахунку міцності конструкцій»

Виконав студент гр. ІС-82мп Марочканич Олександр Романович

Керівник Хіміч Олександр Миколайович

СХЕМА СТРУКТУРНА ПОСЛІДОВНОСТІ



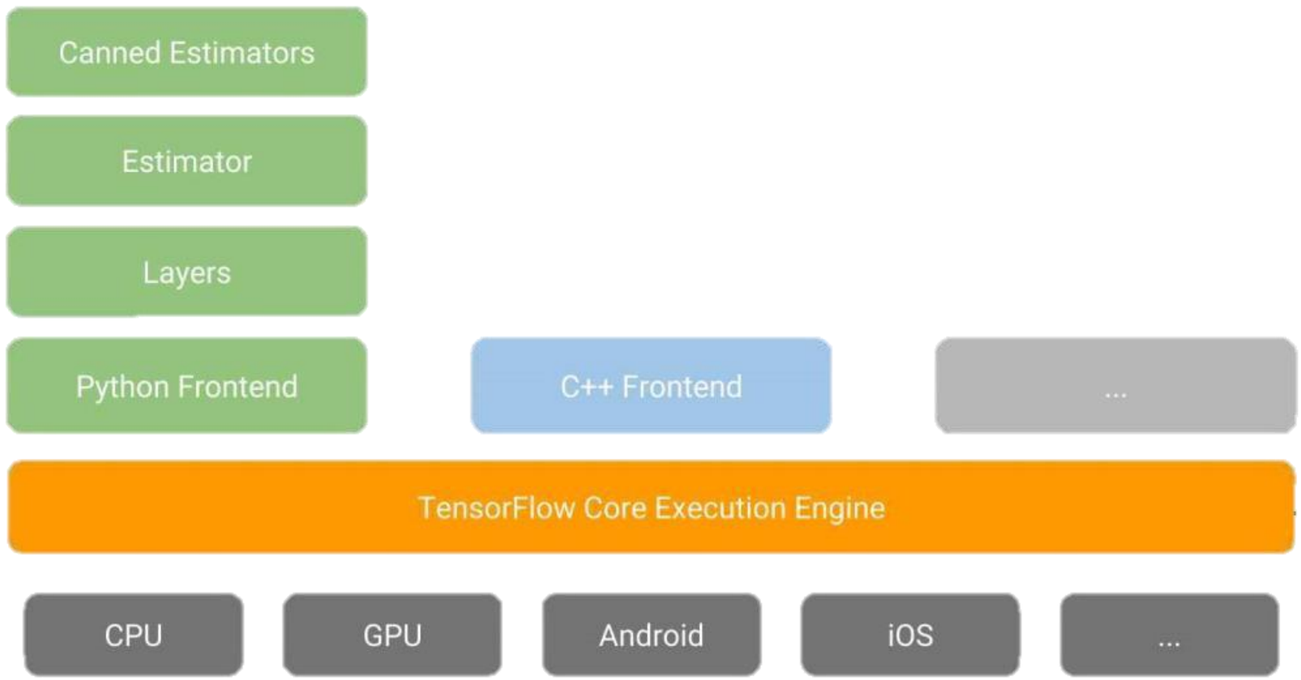
Демонстраційний плакат до магістерської дисертації

на тему «Інтелектуалізація обчислень для задач розрахунку міцності конструкцій»

Виконав студент гр. ІС-82мп Марочканич Олександр Романович

Керівник Хіміч Олександр Миколайович

СХЕМА СТРУКТУРНА КОМПОНЕНТІВ



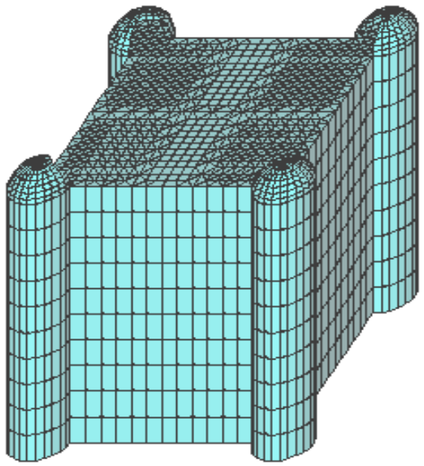
Демонстраційний плакат до магістерської дисертації

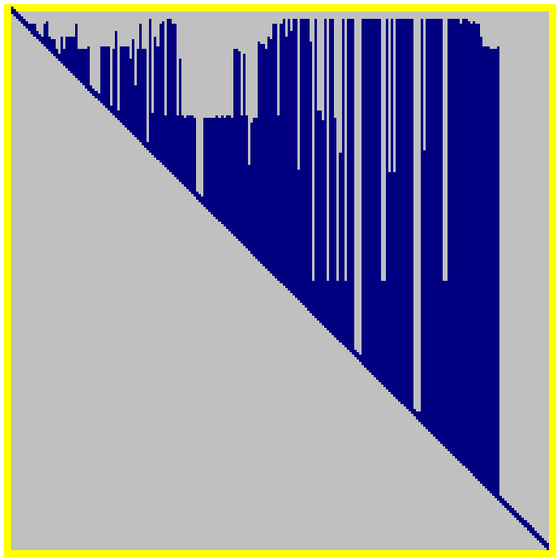
на тему «Інтелектуалізація обчислень для задач розрахунку міцності конструкцій»

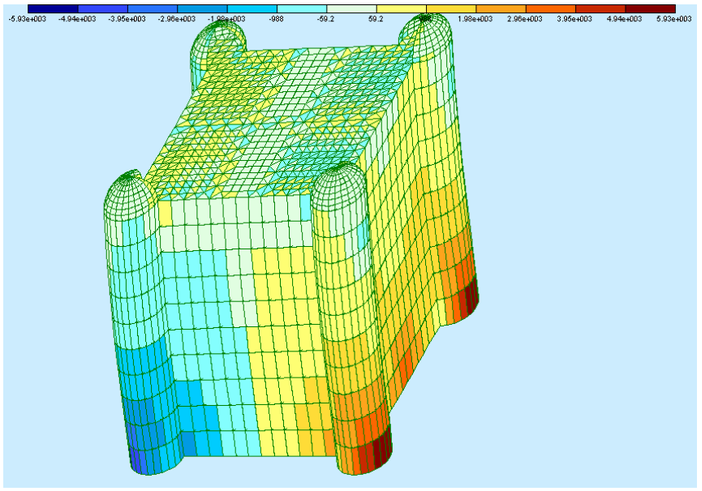
Виконав студент гр. ІС-82мп Марочканич Олександр Романович

Керівник Хіміч Олександр Миколайович

ДЕМОНСТРАЦІЯ СТАТИЧНОГО РОЗРАХУНКУ НАПРУГО-ДЕФОРМОВАНОГО СТАНУ ПРОМИСЛОВОЇ СПОРУДИ







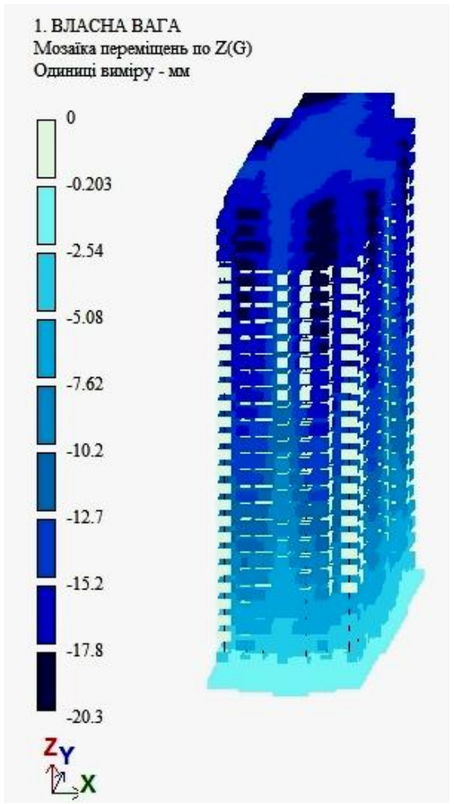
Демонстраційний плакат до магістерської дисертації

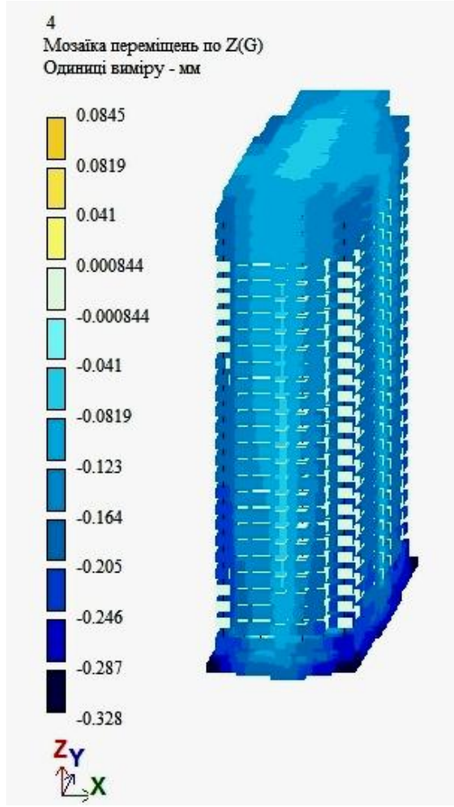
на тему «Інтелектуалізація обчислень для задач розрахунку міцності конструкцій»

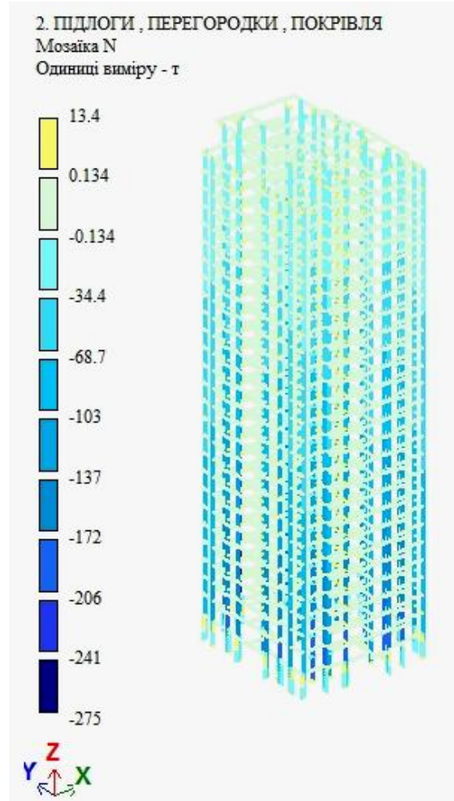
Виконав студент гр. ІС-82мп Марочканич Олександр Романович

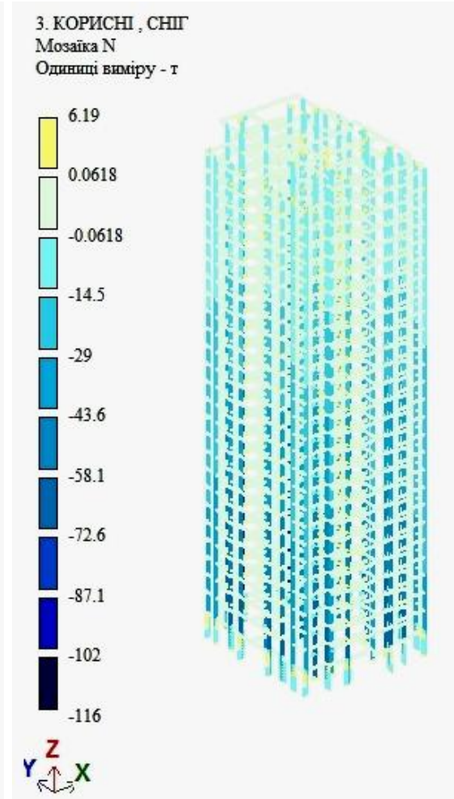
Керівник Хіміч Олександр Миколайович

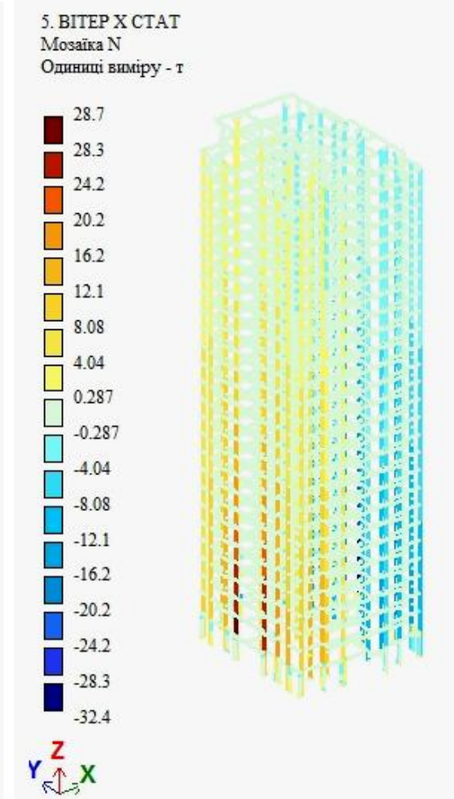
ДЕМОНСТРАЦІЯ СТАТИЧНОГО РОЗРАХУНКУ НАПРУГО-ДЕФОРМОВАНОГО СТАНУ ЖИТЛОВОЇ БУДІВЛІ











Демонстраційний плакат до магістерської дисертації

на тему «Інтелектуалізація обчислень для задач розрахунку міцності конструкцій»

Виконав студент гр. ІС-82мп Марочканич Олександр Романович

Керівник Хіміч Олександр Миколайович